HAAGA-HELIA
ammattikorkeakoulu

Sovelluksen käyttöliittymän tyylioppaan toteutus. Case Richen Ideapp.

Julius Juntunen

Opinnäytetyö
Tietojenkäsittely
2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

20.1.2015

Tekijä(t)
Julius Juntunen

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön otsikko
Sovelluksen käyttöliittymän tyylioppaan toteutus. Case Richen Ideapp.

Sivu- ja liitesivumäärä
52 + 8

Opinnäytetyön otsikko englanniksi
Creating UI style guide to application. Case Richen Ideapp.

Tyyliopas voi tarkoittaa monenlaista ja monen näköistä dokumenttia, jonka sisältö määräytyy usein sitä käyttävän yrityksen toimialan mukaan. Usein tyyliopas on tarkoitettu vain yrityksen sisäiseksi oppaaksi eikä sitä haluta jakaa julkiseksi, mutta jotkin yritykset ovat julkaisseet tyylioppaansa Internetissä. Tyyliopas voi sisältää esimerkiksi yrityksen logon käytön ohjeita sekä tekstin ja julkisten julkaisuiden äänenpainoon ja fonttiin ohjeita yrityksen työntekijöille. Tietotekniikassa tyyliopas voi puolestaan sisältää sovelluksen tai sivun ulkoasun komponentit tai ohjeita sovelluskehittäjille sovelluskehityksessä käytettävän kieleen. Tyylioppaissa on siis toimialakohtaisia eroavaisuuksia. Tässä opinnäytetyössä keskitytään sovelluksen ulkoasun tyylioppaaseen ja sen luomiseen toimeksiantajan sovellukseen.

Opinnäytetyön toimeksiantajan yritys Richen Oy on kokeillut ulkoasun tyyliopasta eräässä asiakkaalleen tekemässä projektissa ja todenneet tyyliopas-vetoisen kehittämisen helpottavan heidän työtään ja olevan hyödyllinen työkalu yhdistämään kehitystiimiään. Tästä johtuen heille muodostui tarve ulkoasun tylioppaan toteuttamiseen Ideapp-tuotteeseensa. Tämän opinnäytetyön tarkoitus on luoda Richenille käyttöönotettava tyyliopas ja samalla valmistella sovellus tyyliopas-vetoista kehittämistä varten järjestelemällä sovelluksen tyylitiedostot uudella tavalla.

Opinnäytetyössä esitellään mikä on tyyliopas, kuka sellaista käyttää sekä mitä mahdollisia hyötyjä ja haittoja tällaisella toimintatavalla on. Lisäksi esitellään KSS, joka on työkalu tylioppaan luomiseen, ja toteutetaan Ideapp:n tyyliopas tällä. Toteutus-osiossa esitellään mitä muutoksia sovellukseen ja sen tyylitiedostoihin tulee tehdä, jotta saavutetaan haluttu lopputulos. Myös opinnäytetyöstä syntynyt tyyliopas esitellään ja annetaan yritykselle jatkoa varten ehdotuksia tylioppaan käyttöön.

Opinnäytetyöhön oli alun perin varattu 14 viikkoa aikaa, mutta projektin aikana alkanut työ hidasti projektia ja aiheutti kolmen viikon myöhästymisen alkuperäisestä aikataulusta. Projekti saatiin kuitenkin vietyä loppuun ja tuloksena syntyi Richenille käyttöönotettava ulkoasun tyyliopas Ideapp-sovellukseen.

Asiasanat
Tyyliopas, KSS, Käyttöliittymä

HAAGA-HELIA
University of Applied Sciences

Abstract

20.1.2015

Author(s)
Julius Juntunen

Degree programme
Degree Programme in Business Information Technology

Report/thesis title
Creating UI style guide to application. Case Richen Ideapp.

Number of pages and appendix pages
52 + 8

A style guide can mean many different kinds of things depending on the branch of business of the company using it. A style guide is usually created only for internal use. There are however few companies, that publish their style guide on the internet. A style guide can contain for example guidelines for the use of a company logo in documentations or guidelines for font and tone of voice for public announcements. In computer science, a style guide can be created for developers as guidelines for coding. A UI style guide contains user interface elements and components from an application or a website. In this thesis we are focusing on UI style guide and how to create one for an application of a specific company.

The subject of this thesis is a company named Richen Oy. They have used style guides in their projects to customers and they have decided that their own application also needs a style guide. They have noticed that style guide-driven developing is helping and making their developing easier so they wanted a style guide for their Ideapp-application. The purpose of this thesis is to create UI style guide for Richen and at the same time prepare their developing work for style guide-driven developing by arranging stylesheets the way they need to be for making that possible.

The thesis explains what a style guide is and who uses them. Also is mentioned what kind of benefits and disbenefits style guide may have. KSS, the style guide creation tool, is shown and the style guide is created by using it. In Creating-section is shown what changes is needed to do to stylesheets so the wanted result is achieved. At the end is shown the created UI style guide for Ideapp and some suggestions is given to Richen for future use of style guide.

In first place there were planned 14 weeks for the thesis but due to a new job started in the middle of the thesis project, the schedule was delayed by three weeks. Despite that, the thesis was completed and the result is a working UI style guide for Richen´s Ideapp-application.

Keywords
Style guide, KSS, user interface

Sisällys

1 Johdanto

Tässä opinnäytetyössä esitellään mikä on style guide (jäljempänä tyyliopas) sekä toteutetaan opinnäytetyön toimeksiantajalle tyyliopas heidän selainpohjaiseen sovellukseensa. Opinnäytetyön taustana on opinnäytetyön toteuttavan opiskelijan työharjoittelu Richen Oy:ssä ja tätä kautta saatu aihe käyttöliittymän tyylioppaan toteuttamisesta yrityksen omaan sovellukseen. Projekti sai alkunsa siitä, että yrityksellä oli tarve yksinkertaistaa ja helpottaa heidän Ideapp -tuotteensa kehitystä. Richen on yrityksenä nuori pelillistämiseen erikoistunut pienyritys, joka toteuttaa myös muitakin digitaalisia projekteja asiakkailleen. Heidän oma tuotteensa on Ideapp, joka on yrityksille ja yhteisöille tarkoitettu ideointityökalu, jonka tarkoitus on helpottaa yrityksen sisäistä ideointia ja tuoda yrityksen työntekijät ja johtohenkilöt samalle viivalle luovan ideoinnin mahdollistamiseksi.

Richen on käyttänyt tyylioppaita muutamassa asiakkailleen tekemissään projekteissa, joten he ovat todenneet tyylioppaan käytön toimivaksi toimintatavaksi helpottamaan ja yksinkertaistamaan tuotteen tai palvelun kehitystä. Tästä johtuen he ovat päätyneet siihen, että heidän Ideapp -sovelluksensa tarvitsee myös oman tyylioppaan. Projekti tulee myös edesauttamaan Richenin työntekijöitä kohti tyyliopas-vetoista sovelluskehitystä, joka on yhtenä tarkoituksena tyylioppaan luomisessa ja käyttöönotossa. Tyyliopas vaatii sovelluksen tyylitiedostojen uudelleenmuokkaamista sekä uusien tyylitiedostojen luomista. Tätä tyylitiedostoissa tapahtuvaa muutosta tullaan esittelemään tässä opinnäytetyössä.

Opinnäytetyön tavoitteeksi asetettiin ensisijaisesti toimivan tyylioppaan toteuttaminen toimeksiantajalle heidän Ideapp-sovellukseensa. Opinnäytetyön tavoitteena on myös esitellä mikä on tyyliopas ja varsinkin käyttöliittymän tyyliopas, sekä tuoda esille mitä hyötyjä ja millaisia ongelmia kyseisellä toimintatavalla mahdollisesti on. Opinnäytetyössä annetaan myös muutama esimerkki yritysten tyylioppaista ja esitellään tyylioppaan toteuttamiseen käytettäviä työkaluja. Näistä yhdellä työkalulla tullaan myös toteuttamaan Richenin tyyliopas. Toteutus-osiossa perehdytään toimeksiantajalle toteutettavan tyylioppaan toteuttamiseen ja dokumentoidaan tämä.

Opinnäytetyö ei ota kantaa siihen onko tyylioppaan käyttö jokaisessa tilanteessa se paras tai tehokkain ratkaisu, vaan se pyrkii ennemmin esittelemään aiheen ja tyylioppaan tekemiseen käytettäviä työkaluja sekä esimerkkityönä toteuttaa toimiva ja käyttöönotettava tyyliopas toimeksiantajan yritykselle.

Sanasto

Esiprosessori

Esimerkiksi LESS on dynaaminen tyylitiedoston kieli, joka mahdollistaa muun muassa muuttujien ja eri funktioiden käytön, joita normaali CSS-tyylitiedosto ei tue. Esimerkkinä LESS muuttujasta on ”@nice-blue: #5B83AD;”, jota voidaan käyttää normaaliin tapaan vaikkapa otsakkeen värin määrityksessä seuraavasti:

```
#header {
        color: @nice-blue;
       }
```

Normaalisti CSS-tyylitiedostossa edellä oleva tulisi merkitä siten, että väriin laitetaan halutun värin heksakoodi kuten #5B83AD.

Grunt / Gulp

Automatisointityökaluja joilla voidaan hoitaa toistuvia tehtäviä, jotta niitä ei kehittäjän tarvitse erikseen itse tehdä. Esimerkiksi less-tiedoston kääntäminen css-tiedostoksi onnistuu automatisointityökalulla.

Hogan

Mallipohjatyökalu, joka mahdollistaa muun muassa muuttujien käytön.

Mallipohja (Template)

Komponenteista ja rakenteista koottu malli sivusta ja sen ulkonäöstä.

Komponentti (Component)

Pienin osa tyyliä/ sivua. Esimerkiksi painike tai input-kenttä.

Pseudo-luokka (Pseudo-class)

Käytetään määrittelemään elementin tiettyä tilaa. Esimerkiksi pseudo-luokka :hover luo tietyn efektin kun hiiren vie elementin päälle, jolla on pseudo-luokkana :hover.

Rakenne (Structure)

Kahden tai useamman komponentin yhdistelmä. Esimerkiksi otsake eli header tai lomake.

2 Tyylioppaan teoriaa

Tässä teoriaosuudessa perehdytään siihen, mikä on tyyliopas ja kuka sellaista käyttää. Osiossa annetaan myös muutama esimerkki tyylioppaasta sekä kerrotaan mitä tarvitaan tyylioppaan luomiseen ja millä sellaisen voi toteuttaa.

2.1 Mikä on tyyliopas?

Tyylioppaita tarkastellessa vastaan tulee monia eri versioita tyylioppaista. Toimialaan liittyen ne voivat olla hyvinkin erilaisia niin sisällöltään kuin ulkoasultaan ja toteutustavaltaan. Tyyliopas voi olla toteutettu tulostettavaksi ja jaettavaksi pdf-muotoiseksi esitykseksi tai vaihtoehtoisesti se on toteutettuna Internetiin, josta siihen on helppo kaikkien sitä tarvitsevien päästä käsiksi. Internetiin toteutettua tyyliopasta on luonnollisesti helpompi pitää ajantasaisena kuin tulostettua versiota, koska sitä ei tarvitse tulostaa ja jakaa uudelleen sitä käyttäville. Internetissä oleva tyyliopas voidaan myös asettaa salasanan taakse, jolloin sitä pääsevät tarkastelemaan vain halutut henkilöt, mikäli tyyliopasta ei haluta laittaa julkiseksi kaikkien nähtäville.

Style guide eli tyyliopas voi tarkoittaa monta asiaa ja kontekstista riippuen pitää sisällään eri asioita. Kat Nevillen artikkeli Designing Style Guidelines For Brands And Websites esittelee monipuolisesti mitä erilaiset tyylioppaat voivat pitää sisällään (Neville 2010). Esimerkiksi kirjoittajille tai toimittajille tyyliopas voi tarkoittaa puhtaasti ohjeita käytettävästä kielestä ja yleisistä kirjoitusohjeista tietyssä julkaisussa. Jonkin brändin tai tuotemerkin tyylioppaaseen voidaan sisällyttää äänen paino, jolla tulee kirjoittaa sekä logon käyttö (Neville 2010). Tietotekniikassa tyyliopas voi tarkoittaa muun muassa kehittäjille suunnattuja ohjeita ja määrittelyitä koodin kirjoitukseen ja syntakseihin sekä koodin kommentointiin. Tyyliopas voi pitää sisällään myös sivun rakenteen ja komponentit listattuna sekä näiden havainnollistavan esimerkin ja html-koodin, jolla kyseisen komponentin saa toteutettua sivulle. (Braun 2013.)

Tyylioppaita on myös erilaisia toteutuksia kuten staattinen tyyliopas ja elävä tyyliopas. Nämä eroavat toisistaan muun muassa toteutustavoiltaan sekä ylläpidettävyydeltään. Käyttötarkoitukseltaan ne toimittavat kuitenkin samaa asiaa eli esittelevät jonkin asian toteutuksen ohjeet.

2.1.1 Static style guide eli staattinen tyyliopas

Staattinen tyyliopas tarkoittaa sitä, että tyyliopasta ei ole suoraan linkitetty sen sovelluksen tai Internetsivun koodiin, jonka ulkoasukomponentit se pitää sisällään, vaan se on

erillinen esitys näistä komponenteista. Toisin sanoen tyyliopas on tällöin vain oma kokonaisuutensa erillään tuotannossa olevasta koodista. Koodista on siis kaksi erillistä identtistä dokumenttia, tyyliopas ja itse tuotannossa tai käytössä oleva tiedosto tai tiedostot. Tämä johtaa siihen, että aina jos sivun tai sovelluksen HTML:ään tai tyylitiedostoon tekee muutoksia, joutuu myös tekemään samat muutokset manuaalisesti tyylioppaaseen, jotta se pysyisi ajantasaisena. Staattinen tyyliopas saattaakin näin ollen jäädä kehityksessä jälkeen jos sitä ei muisteta päivittää. Mikäli tyyliopas ei ole enää ajantasainen vaan se on jäljessä toteutetusta koodista, se on menettänyt merkityksensä. (Feather 2014.).

Staattista tyyliopasta voidaan pitää parempana vaihtoehtona kuin elävää tyyliopasta esimerkiksi pelkästään komponenttien esittelemiseen vaikkapa sovelluksen tai sivun kehityksessä mukana olevalle kolmannelle osapuolelle tai asiakkaalle. Näin voidaan toimia varsinkin jos ei haluta esitellä vielä kehityksessä olevia komponentteja.

2.1.2 Living style guide eli elävä tyyliopas

Toisin kuin staattinen tyyliopas, elävä tyyliopas on suoraan kytköksissä kehityksessä käytettävään tyylitiedostoon, josta se generoidaan automaattisesti. Tämä tarkoittaa siis sitä, että tyyliopas on, tai sen ainakin pitäisi olla, koko ajan ajantasainen ja sisältää samat komponentit kuin kehityksen tyylitiedosto, koska tyyliopas on suoraan luotu siitä. Elävässä tyylioppaassa on juuri ne elementit ja komponentit, jotka juuri sillä hetkellä ovat käytössä sovelluksessa tai sivulla. (Feather 2014.)

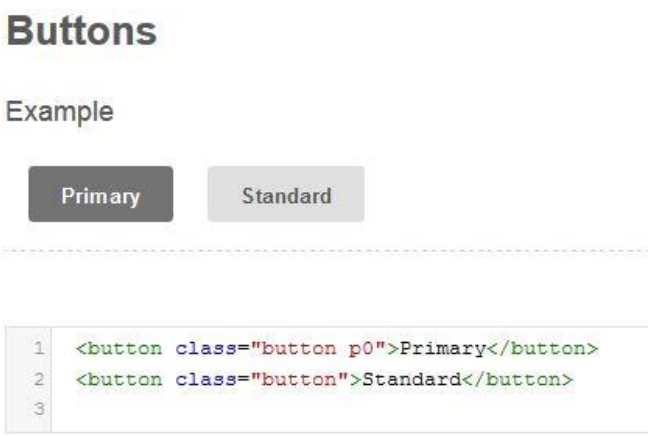
Elävän tyylioppaan käyttö on vasta viimevuosina alkanut yleistymään, tai ainakaan sitä koskevat artikkelit ja kirjoitukset, jotka opinnäytetyötä tehdessä löytyivät, ovat suurimmalta osin vanhimmillaan vain muutaman vuoden vanhoja.

Elävää tyyliopasta eli living style guide voisi pitää suositellumpana tapana toteuttaa käyttöliittymän tyyliopas kuin staattista tyyliopasta, sillä se ei jää, tai ainakaan ei pitäisi jäädä, jälkeen kehityksessä samalla tavalla kuin staattinen tyyliopas, jota ei välttämättä muisteta päivittää vastaamaan tuotantoa.

2.2 Käyttöliittymän tyyliopas

Ulkoasuun liittyvää tyyliopasta, joka pitää sisällään muun muassa sovelluksen tai sivun rakenteet ja komponentit sekä fontit ja värit, kutsutaan nimellä UI style guide eli käyttöliittymän tyyliopas. Rakenteilla tarkoitetaan kahden tai useamman komponentin yhdistelmää. Näitä ovat esimerkiksi sivun otsake ja lomakkeet. Komponentteja puolestaan ovat esimer-

kiksi painikkeet ja sivun syöttökentät. Komponentit ovat samalla rakenteiden ”rakennuspalikoita”, joista rakenteet rakentuvat.



Kuva 1. Esimerkki painike komponentista MailChimp.com sivun tyylioppaasta (MailChimp 2014)

Kuvassa yksi on esitelty MailChimp -sähköpostiohjelman tyylioppaan painike-komponentti. Tyyliopas rakentuu ensin esimerkistä millainen painike on ja tämän alla on koodiin upotettava html-koodi, jolla painikkeen saa luotua sovellukseen tai sivuun. Kuvassa näkyvän painikkeen saa toteutettua antamalla button-elementille button -nimisen luokan, joka sisältää painikkeen tyylimääritteet. Kuvan Primary-painikkeella on myös luokkana p0, joka tässä tapauksessa tuo painikkeelle normaalia tummemman taustavärin.

Kuva 2. Esimerkki input-kenttä komponentista MailChimp.com sivun tyylioppaasta (MailChimp 2014)

Kuvassa kaksi on MailChimp -sähköpostiohjelman tyylioppaasta otettu text input -kenttä. Kuten edellisessäkin painike esimerkissä, on tässäkin esimerkkinä olevan input -kentän alla HTML-koodi, jolla kentän saa helposti kopioitua paikkaan, jossa sitä tarvitsee.

Tarkoituksena on siis se, että kopioimalla tyylioppaasta esimerkkinä olevan elementin tai rakenteen alla olevan html-koodin ja liittämällä sen rakenteilla olevaan sivuun tai sovellukseen, saa tähän rakennettua helposti ja nopeasti haluamansa komponentin tai rakenteen. Toimimalla näin kehittäjän ei tarvitse alkaa miettiä tyylimäärittelyitä komponentille, koska ne ovat jo valmiiksi määriteltyinä ja tarvitsee vain niin sanotusti ”käydä hakemassa” haluttu komponentti tyylioppaasta. Hyvin toteutettuna komponentti on samanlainen riippumatta siitä mihin kohtaan sivua ja sen rakennetta komponentin toteuttaa.

Itse tyylitiedoston määrittelyitä ei välttämättä sisällytetä tyylioppaaseen nähtäville, vaan siihen riittää pelkästään komponentit ja niiden HTML-koodi. Halutessaan tyylioppaaseen saa sisällytettyä myös CSS-määritteet näkyville ja esimerkiksi StyleDoccoa käyttäessä ne listautuvat automaattisesti tyylioppaaseen. Useimmissa tarkastelluissa tyylioppaissa ei kuitenkaan oltu liitetty tyylimääritteitä nähtäville. Esimerkiksi Githubin ja Lonely Planetin tyylioppaissa ei ollut näitä määritteitä näkyvillä lainkaan.

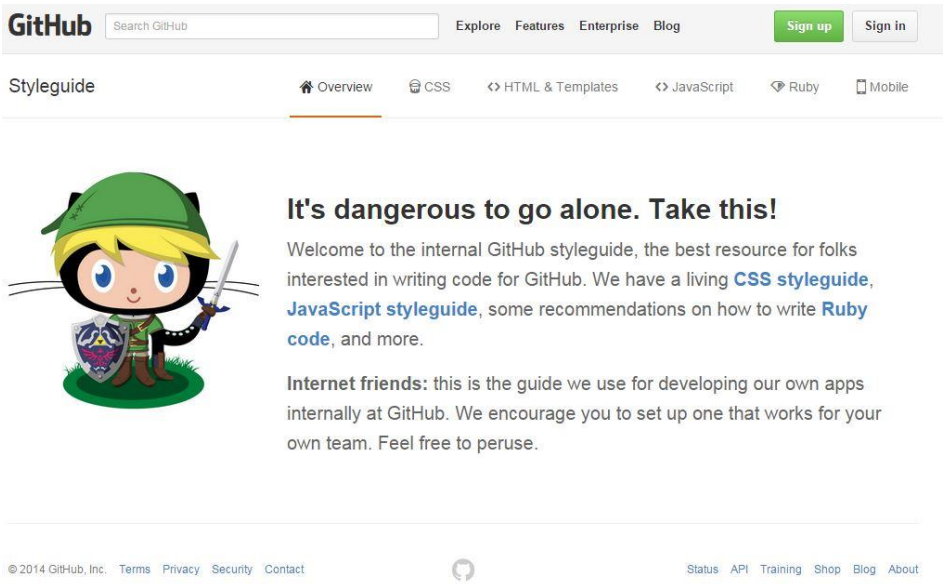
2.3 Kuka käyttää?

Tyylioppaita käytetään varsinkin suuremmissa yrityksissä varmasti paljon, mutta ne voivat olla vain yrityksen tavaramerkkiä koskevia tyylioppaita, joissa on määriteltynä esimerkiksi kuinka yrityksen logo tulee sijoittaa vaikkapa pipoon tai kuinka paljon tyhjää tilaa logon ympärillä tulee olla (Sullivan 2013, 4/83). Yksittäisiä yrityksiä, jotka käyttävät käyttöliittymän tyyliopasta, on vaikea mennä varmasti nimeämään, sillä tyylioppaat ja varsinkin koodin mukana elävät tyylioppaat on usein tarkoitettu vain ainoastaan yrityksen työntekijöiden nähtäville. Uskoisin kuitenkin, että käyttöliittymän tyylioppaita käytetään nykyään varsinkin isoimmissa sovelluksissa ja sivuissa tai siihen ollaan mahdollisesti siirtymässä. Yritysten käyttöliittymien tyylioppaita ei useinkaan laiteta julkiseen jakoon kaikkien nähtäville, mikä vaikuttaa niiden saatavuuteen.

Osa yrityksistä on kuitenkin laittanut käyttöliittymän tyylioppaansa nähtäville Internetiin. Tällaisia yrityksiä ovat muun muassa Starbucks, Lonely Planet, BBC ja Trulia. Suomalaisista yrityksistä muun muassa Veikkaus on laittanut käyttöliittymän tyylioppaansa julkiseksi (Veikkaus tyyliopas 2014).

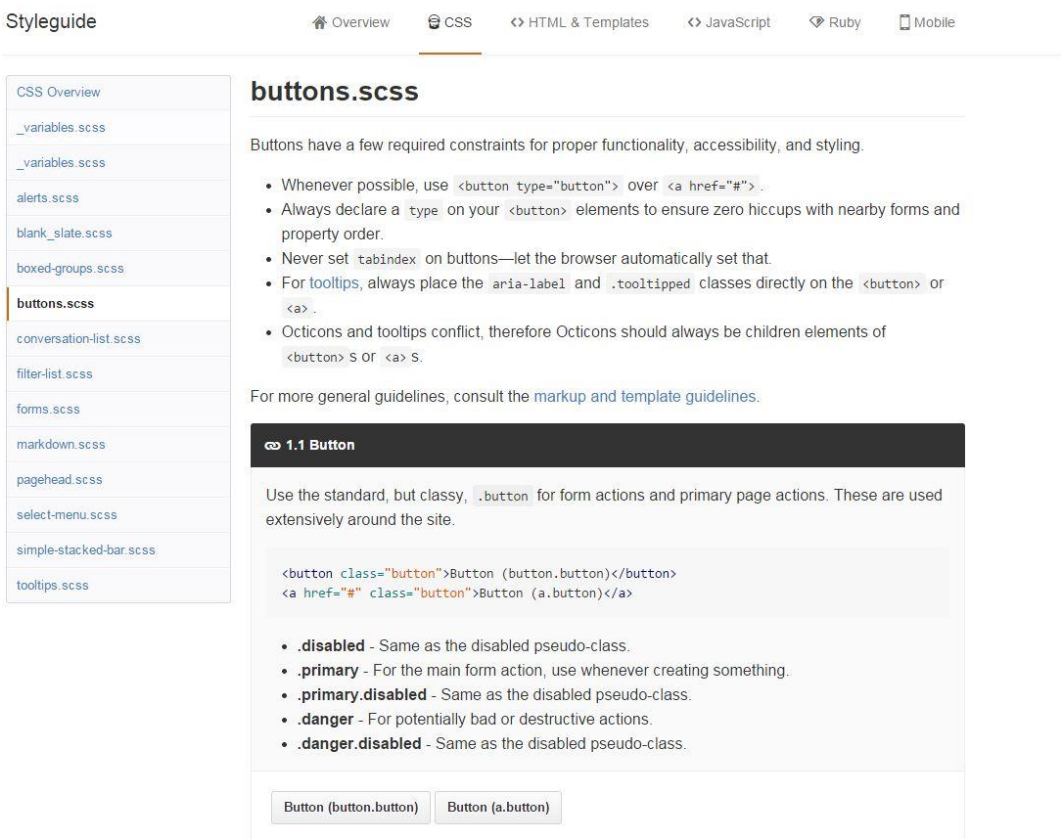
2.3.1 GitHub

GitHub on verkossa toimiva versionhallintapalvelu, jonka avulla pystyy hoitamaan projektinsa versionhallinnan. Se tarjoaa myös maailman laajimman open source yhteisön mahdollisuuden eli mahdollisuuden jakaa projektinsa vapaasti verkossa muiden nähtäville ja kommentoitavaksi. (GitHub 2014.)



Kuva 3. GitHub:n tyyliopas (GitHub 2014)

GitHub:n tyyliopas on kattava kokonaisuus, joka pitää sisällään ohjeet muun muassa CSS:ää, JavaScriptiä ja Rubyä varten. Kuvasta kolme nähdään Github:n tyylioppaan ”etusivu”, joka on rakennettu mukailemaan GitHub sivuston ulkoasua. GitHub:n tyyliopas on tarkoitettu saatetekstin mukaan GitHub:n omien sisäisten sovellusten rakentamiseen, mutta sitä saa halutessaan käyttää vapaasti vaikka esimerkkinä.



Kuva 4. GitHub tyylioppaan painikkeet (GitHub 2014)

Kuvasta neljä on hyvin nähtävillä esimerkiksi GitHub:n tyylioppaan rakenne CSS:ää varten. Oppaan yläosassa on valikko, jossa on muun muassa osiot CSS, HTML & Templates ja Javascript. Github on näin jakanut tyylioppaansa erikseen omiin osuuksiinsa tyylit esimerkiksi Javascript:lle ja CSS:lle. Kuvassa auki olevan CSS -osion vasemmassa reunassa nähdään osion sisäinen sisällysluettelo, jossa on listattu osion sisältö. Auki oleva buttons.scss on nimensä mukaisesti osa, joka sisältää ohjeistuksen painikkeille ja niiden käytölle.

Ensimmäisenä osassa on kerrottu yleisiä ohjeita siitä kuinka painikkeita tulee oikeaoppisesti käyttää GitHub:ssa. Esimerkkinä tyylioppaassa kehotetaan käyttämään aina kun mahdollista mieluummin <button type=”button”> kuin <a href=”a”>. Seuraavaksi tyylioppaassa on kerrottu kuinka painike rakentuu koodillisesti sekä mitä lisäluokkia tai määrittelyitä painikkeelle voi antaa sekä näiden selitykset mitä ne tarkoittavat. Esimerkiksi disabled -luokka on oppaan mukaan sama kuin pseudo-luokka disabled eli ”laitettu toimintakyvyttömäksi”, jolloin painike ei ole käytössä.

2.3.2 Lonely Planet

Lonely Planet on yritys, joka julkaisee matkaopaskirjoja sekä muuta matkailuun liittyvää digitaalista sisältöä. Lonely Planet on myös yksi niistä yrityksistä, jotka ovat julkaisseet käyttöliittymän tyylioppaansa ja sitä voi käydä tarkastelemassa heidän sivuiltaan.

Ian Feather kirjoittaa blogikirjoituksessaan A Maintainable Style Guide (Feather 2014) työstään Lonely Planetin tyylioppaan kanssa. Feather kirjoittaa idean tyylitiedoston tekemiseen lähteneen tarpeesta selkeyttää ja purkaa sen hetkistä käyttöliittymää, joka oli kasvanut liian suureksi ja monimuotoiseksi. Lonely Planetin ongelmaksi oli tullut liian monien erilaisten mallipohjien olemassa olo ja yhteisen tyylin ja rakenteiden puuttuminen sivustolla, joten he olivat ajautuneet tilanteeseen, jossa he olivat pakotettuja rakentamaan kokonaan uudelleen saadakseen selkeämpi ja yhtenäisempi toteutus.

Feather kirjoittaa tehneensä ensimmäisestä tyylioppaasta staattisen tyylioppaan, joka antoi hieman selkeyttä sen hetkisestä käyttöliittymästä, mutta ei kuitenkaan tuntunut sopivan heidän työnkulkuunsa ja jäikin nopeasti jälkeen päivittämättömänä ylimääräisenä dokumenttina.Toinen versio oli vuonna 2012 elävä tyyliopas, joka oli tehty suositulla KSS -työkalulla. Featherin mukaan ei kuitenkaan mennyt kuin muutama kuukausi kun tyyliopas oli jo erkaantunut heidän sovelluksensa komponenteista. Syy erkaantumiseen Featherin mukaan löytyy siitä, kuinka nykyiset tyylioppaan tekemiseen tarkoitetut työkalut toimivat.

Tyylioppaan luomiseen tarkoitetut työkalut kuten KSS toimivat siten, että tietääkseen kuinka komponentti on rakennettu ja renderöidäkseen sen tyylioppaaseen, ne analysoivat tyylitiedostoa ja jäsentelevät sen kommentteja, joista työkalu tietää luoda tyylioppaan. Featherin mukaan ongelma piilee siinä, että tyylioppaaseen luotu mallipohja ei ole sama kuin se mallipohja, jota he käyttävät tuotannossa vaan parhaimmillaankin se on suora kopio. Tämä taas tarkoittaa sitä, että on kaksi erillistä mallipohjaa, joita täytyy päivittää erikseen. Featherin mukaan mallipohjien kopiot osoittautuvat ongelmaksi varsinkin kun sovellus laajenee ja mallipohjien lukumäärä lisääntyy.

Toinen ongelma Featherin mukaan liittyy mallipohjien jakaantumiseen. Hänen mukaansa vaikka tyyliopas olisikin ajantasainen, ongelma ilmenee siinä vaiheessa kun siihen tehdään muutoksia ja muutos pitäisi kopioida mallipohjiin. Featherin mukaan komponentin tekijä ei tiennyt enää heidän tapauksessaan, missä kaikissa mallipohjissa komponentti oli käytössä. Saattoi myös olla niin, että komponentin tekijä ei ollut edes tietoinen jostakin sovelluksesta, jossa komponentti oli käytössä. Feather kirjoittaakin, että tämä johti Lonely

Planetilla siihen, että heidän koodinsa ei enää ollut uudelleenkäytettävää vaan he löysivät käyttävänsä taas kahdennettuja komponentteja ja heidän koodinsa paisui suureksi.

Ratkaisuksi Lonely Planetin ongelmaan löytyi uudenlaisen ajattelun omaksuminen. Featherin mukaan tyylioppaan tulisi keskittyä mallipohjiin ja sovelluksen mallipohjien tulisi olla samat kuin mistä tyyliopas renderöidään. Lonely Planet muokkasi sovelluksensa rakentumaan moduuleista. He eristivät käyttöliittymästään osia, purkivat ne pienemmiksi osiksi ja tekivät niistä komponentteja. Nyt heidän tyylioppaansa ja sovelluksena käyttävät saman komponenttikerroksen komponentteja ja tyyliopas pysyy automaattisesti ajantasaisena, koska se on osa sovellusta.

2.3.3 Trulia

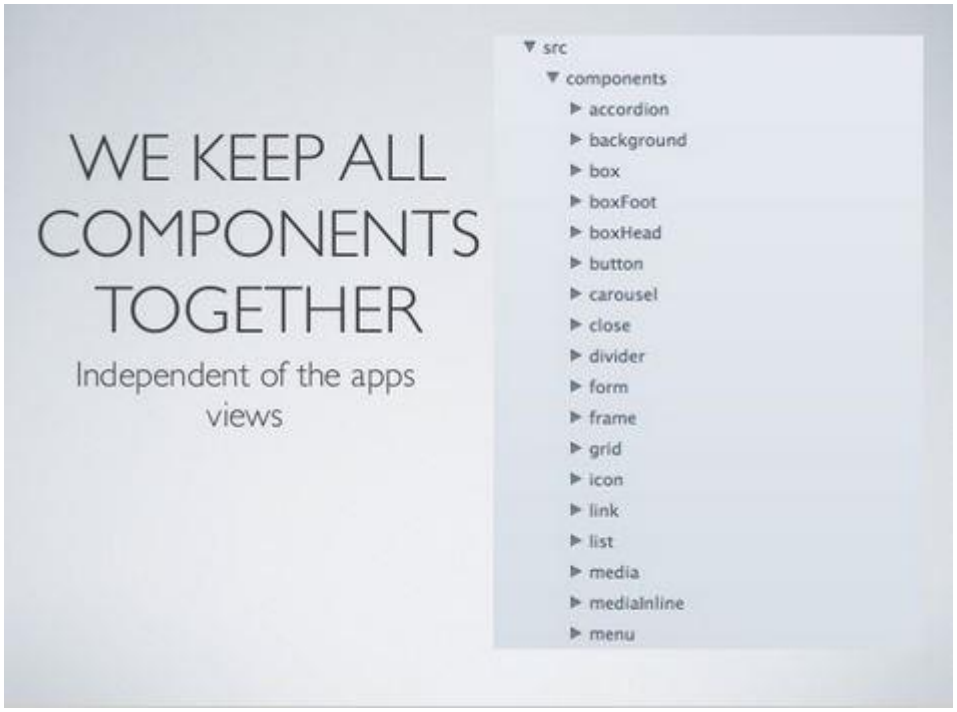
Trulia on Yhdysvaltalainen kiinteistöfirma, jonka on perustanut Perttu Pitkäsen Digitoday:ssa 18.11.2014 julkaistun artikkelin mukaan suomalainen Sami Inkinen vuonna 2005 (Pitkänen 2014.). Trulialle tyyliopasta tekemään palkattiin Nicole Sullivan, joka on Pivotal Labs:n toimitusjohtaja sekä CSS ja suorituskyky insinööri.

Sullivan esiintyi 2013 vuoden JSConfUS- tapahtumassa (Sullivan 2013, [JSConfUS 2013].), jossa hän puhui työstään Trulian tyylioppaan tekemisessä. Sullivanin mukaan Trulian tavoitteena oli uudistaa heidän Internetsivujensa hakutulos-sivu, koska se kaipasi uudelleen toteuttamista ollakseen selkeämpi ja yhtenäisemmän näköinen muun sivuston kanssa. Sullivanin esityksessään esittämät tulokset tyylioppaan tuomista hyödyistä Trulialle ovat hienoja todisteita siitä, että hyvin toteutettuna tyyliopas antaa lisäarvoa projektiin ja helpottaa sekä parantaa projektin, Internetsivun tai sovelluksen kehittämistä ja toimintaa. Puheessaan Sullivan kertoo kuinka aloitti tyylioppaan teon tutkimalla Trulian sen hetkisiä sivuja ja keräämällä tietoa muun muassa siitä montako eri fonttia ja väriä sivuilla oli käytettävissä, jotta hän pystyi helpottamaan Trulian suunnittelijoita päättämään mitkä värit ja fontit sekä kirjasinkoot tulisi mahdollisesti karsia pois yhtenäisemmän tyylin mahdollistamiseksi. Sullivan kertoo myös koonneensa esimerkkinä painikkeet Trulian sivuilta, jotta niistä voitaisiin miettiä yhtenäisempi tyyli ja vähentää variaatioita. Sullivan kehottaakin tekemään näin kaikille komponenteille.

Sullivanin mukaan CSS aiheuttaa muutamia haittoja joissa tyyliopas voi auttaa. Hänen mukaansa CSS esimerkiksi hidastaa sivun progressiivista latausta eli sitä tilannetta kun selain siirtyy sivulle ja selaimen tausta on valkoinen ennen kuin haluttu sivu latautuu ja tulee näkyviin. Sivujen taustakuvat ovat myös yksi asia, jotka tulevat tyylitiedoston kautta ja niiden latautumiseen voidaan vaikuttaa myös Sullivanin mukaan. Sullivan listaa CSS:n

myös hidastavan javascriptiä ja sen olevan suoraan kytköksissä renderöinti nopeuteen. Hänen mukaansa javascriptin hidastuminen näkyy esimerkiksi jotain painettaessa, jolloin painalluksen vaikutus ei ole aivan välitön vaan hieman hitaahko. (Sullivan 2013, [JSConfUS 2013].)

Tyyliopasta tehdessään Sullivan tiimeineen järjestelivät jokaisen komponentin omaan kansioonsa komponentti-kansion alle (kuva 5), jotta hakemisto pysyisi helppolukuisena ja ymmärrettävänä sekä kaikki tietäisivät mistä mikäkin komponentti löytyy.



Kuva 5. Sullivanin esimerkki hakemistorakenteesta (Sullivan 2013, [JSConfUS 2013], Kalvo 45/83.)

Aivan ongelmitta Sullivanin tyylioppaan tekeminen ei sujunut, sillä hän lähti vanhasta tottumuksesta rakentamaan esimerkiksi välilehtiä liian monimutkaisesti ja monitasoisesti. Tämän huomattuaan Sullivan ymmärsi, että rakennetta oli yksinkertaistettava. Loppuen lopuksi valmis Trulian tyyliopas toi hyviä tuloksia Trulian verkkosivuun, mihin sekä asiakas ja Sullivan olivat erittäin tyytyväisiä. (Sullivan 2013, [JSConfUS 2013].)

Trulian saamia hyötyjä tyylioppaan käyttöön siirtymisestä:

- Uusi HTML-tiedosto on 48% pienempi
- 21% nopeampi sivunlatausnopeus
- 60% nopeampi lataus ensimmäiseen bittiin (eli se aika kun sivu latautuu ja sivu näkyy selaimessa vain valkoisena ja kävijä joutuu odottamaan, että jotain tapahtuisi.)
- Käyttämätön CSS pieneni 251kb:stä 101kb:hen
- Hakutulos-sivun kävijämäärä nousi 11% (Sullivan 2013.)

2.4 Tyylioppaan käytön hyödyt

Kuten Lonely Planetin (Feather 2014) esimerkistä voidaan todeta auttaa tyyliopas selkeyttämään ja yhtenäistämään sovelluksen tai sivun ulkoasua. Myöskin komponenttien ja rakenteiden uudelleenkäytettävyys mahdollistuu mikä puolestaan mahdollistaa muun muassa helpomman jatkokehityksen ja ylläpidettävyyden.

Truliassa (Sullivan 2013) saavutetut positiiviset kokemukset ja tulokset tyylioppaasta ovat myös hyviä esimerkkejä tyylioppaan hyödyistä. Varsinkin sivunlatausnopeuden kasvaminen 21 prosentilla ja 60 prosenttia nopeampi lataus ensimmäiseen bittiin näkyvät suoraan sivua käyttäville asiakkaille sivun nopeampana latautumisena ja nopeampana toimintana. Myöskin uudistetun sivun kävijämäärän nousu on toivottu ja erinomainen saavutettu hyöty.

Luvun kuvankaappukset (kuvat 6,7 ja 8) on otettu Design and Usability Testing Center:n johtajan Chaunsey Wilsonin artikkelista Guidance on Style Guides: Lessons Learned (Wilson 2001). Artikkeliin hyödyt on kerätty Stephen Galenin kirjoittamasta kirjasta A Collaborative Approach to Developing Style Guides (Gale 1995). Kuvissa on taulukkomuodossa tyylioppaan hyötyjä kolmelle eri käyttäjäryhmälle tai taholle. Galen mukaan nämä hyödyt ovat lähtöisin hyvin toteutetusta tyylioppaasta, joka helpottaa ja yksinkertaistaa yhtenäisen ja toimivan kokonaisuuden luomisen. Toki myöskin muu koodaustyö on tehtävä oikein ja kunnolla, jotta esimerkiksi loppukäyttäjien hyödyt tulevat esille.

End Users
Reduced errors
Less frustration
Increased morale
Improved efficiency
Increased confidence
Reduced resistance to new technology

Kuva 6. Tyylioppaan tuomia hyötyjä loppukäyttäjille (Wilson 2001.)

Loppukäyttäjien näkökulmasta (kuva 6) tyylioppaasta voi olla hyötyjä muun muassa siten, että hyvin toteutettu ja toimiva sovellus tai sivu, joka on toimiva ja virheetön, herättää käyttäjissä luottamusta käytettävään palveluun. Hyötyinä on myös se, että toimiva sovellus tai

sivu vähentää vaadittavaa käyttämiseen liittyvää opettelua mikä puolestaan näkyy käyttäjien vähempänä turhautumisena. Nämä kaikki hyödyt ovat Galen mukaan lähtöisin ainakin osittain hyvin toteutetusta tyylioppaasta.

Developers
Maintain control over look and feel
Minimize re-invention
Capitalize on learning
Enable production of reusable software
Reduce development time
Reduce arbitrary design decisions

Kuva 7. Tyylioppaan tuomia hyötyjä kehittäjille (Wilson 2001.)

Galen mukaan hyötyjä kehittäjille (kuva7) on muun muassa se, että ulkoasu ja yleinen tuntuma sovelluksessa tai sivussa pysyy kontrolloitavissa sekä asioiden uudelleen keksimistä ja kehittämistä joutuu tekemään vähemmän. Myös uusien kehittäjien mukaan ottaminen kehitystyöhön on helpompaa ja nopeampaa, koska on antaa selkeät tyylit ja komponentit joilla kehittää sovellusta tai verkkosivua. Myös uudelleenkäytettävyys koodissa kasvaa kun eri komponentit pystytään hyödyntämään useassa eri kohdassa ja voidaan olla varmoja, että lopputuloksena on toimiva ja yleisilmeeseen sopiva komponentti. Uudelleenkäytettävyys esimerkiksi komponenteissa ja rakenteissa vähentää osaltaan kehitykseen käytettävää aikaa, sillä kehityksen tulisi olla melkein kuin kasaisi rakennuspalikoista haluttua lopputulosta. Kehittäjien mahdolliset mielivaltaisesti tehtyjen ulkoasupäätöstenkin tulisi näin ollen vähentyä, kun on antaa selkeät komponentit ja tyylit joita noudattaa ja joilla rakentaa sovellusta tai sivua.

Business Team
Produce usable systems that reduce support costs and increase user satisfaction
Increase market awareness
Increase product awareness
Reduce training costs
Improve staff retention
Increase user acceptance of new systems

Kuva 8. Tyylioppaan hyötyjä liiketoiminnan kannalta (Wilson 2001.)

Liiketoiminnan kannalta (kuva 8) tyylioppaalla voi olla myös osittaisia hyötyjä asiakastyytyväisyyden muodossa, sillä asiakkaat ovat esimerkiksi tyytyväisiä toimivaan ja selkeään sovellukseen, joka on saavutettu käyttäen tyyliopasta kehityksessä. Myös henkilöstön koulutukseen kuluu mahdollisesti vähemmän rahaa, koska tyyliopas antaa helpon ja selkeän tavan ja välineet sovelluksen kehitykseen. Mahdollisuus on myös siihen, että työntekijät ovat tyytyväisempiä ja uskollisempia työnantajalleen eivätkä vaihda työpaikkaa, koska tyyliopas helpottaa heidän työtään antaen selkeän ja yksinkertaisen välineen testata ja toteuttaa uusia komponentteja sovellukseen tai sivuun. (Gale 1995.)

Lähteistä riippumatta tyylioppaan ja varsinkin elävän tyylioppaan käyttäminen sovelluksen tai sivun kehityksessä antaa enemmän hyötyjä kuin haittoja. Tyylioppaan ehdottomiin hyötyihin voidaan ainakin lukea se, että oikein toteutettuna se on tehokas työkalu yhdistämään kehittäjiä ja graafikoita lähempään yhteistyöhön, koska sen avulla pystytään nopeasti esimerkiksi esittelemään uusi komponentti kummallekin osapuolelle. Hyötynä on myös ehdottomasti toimia niin uusien kuin vanhojenkin kehittäjien ”varastona”, josta voi helposti etsiä haluamansa komponentin käyttöön. Varsinkin uusien kehittäjien kohdalla tämä nopeuttaa heidän kouluttamistaan mukaan heille uuden sovelluksen tai sivun kehitykseen.

2.5 Tyylioppaan ongelmia ja riskejä

Tyylioppaat eivät ole täysin ongelmattomia luoda tai ylläpitää. Ian Feather kertoo Lonely Planetin sivuilla julkaisemassaan kirjoituksessaa ”A Maintainable Style Guide” muutamista tyylioppaisiin mahdollisesti liittyvistä ongelmista (Feather 2014). Suurin ongelma Featherin mukaan liittynee varsinkin staattisiin tyylioppaisiin, jotka eivät ole linkitettyinä varsinaiseen kehityksessä käytettävään tyylitiedostoon ja ovat tämän takia alttiita sille, että ne saattavat jäädä helposti jälkeen kehityksestä jos niitä ei päivitetä manuaalisesti. Ongelmaksi saattaa muodostua myös se, että kukaan ei tiedä kenen työtehtävänä on päivittää ja ylläpitää tyyliopasta, mikä johtaa helposti sen vanhentumiseen (Gale 1995).

Riskinä tyylioppaissa on myös se, että niistä saattaa huomaamatta tulla erittäin laajoja kokonaisuuksia, jolloin ne voivat muuttua ennemminkin vaikeakäyttöisiksi ja sekaviksi sen sijaan, että ne helpottaisivat ja nopeuttaisivat kehitystä (Gale 1995). Tämä ongelma saattaa muodostua Galen mukaan esimerkiksi silloin jos yksi tyyliopas on rakennettu pitämään sisällään monen eri sovelluksen komponentit, mikä ei ole suotavaa elleivät sovelluksissa käytettävät komponentit ole täysin samoja.

Epäselvät esimerkit ja viittaukset siihen missä ja miten komponentteja tulee käyttää voivat myös hankaloittaa kehittäjien työtä (Feather 2014). Tyylioppaassa tulisikin olla selkeät merkinnät kuinka ja missä sen komponentteja tulee käyttää, kuten esimerkiksi ”tätä painiketta jossa on ostoskärryikoni, tulee käyttää ostoskorin yhteydessä”.

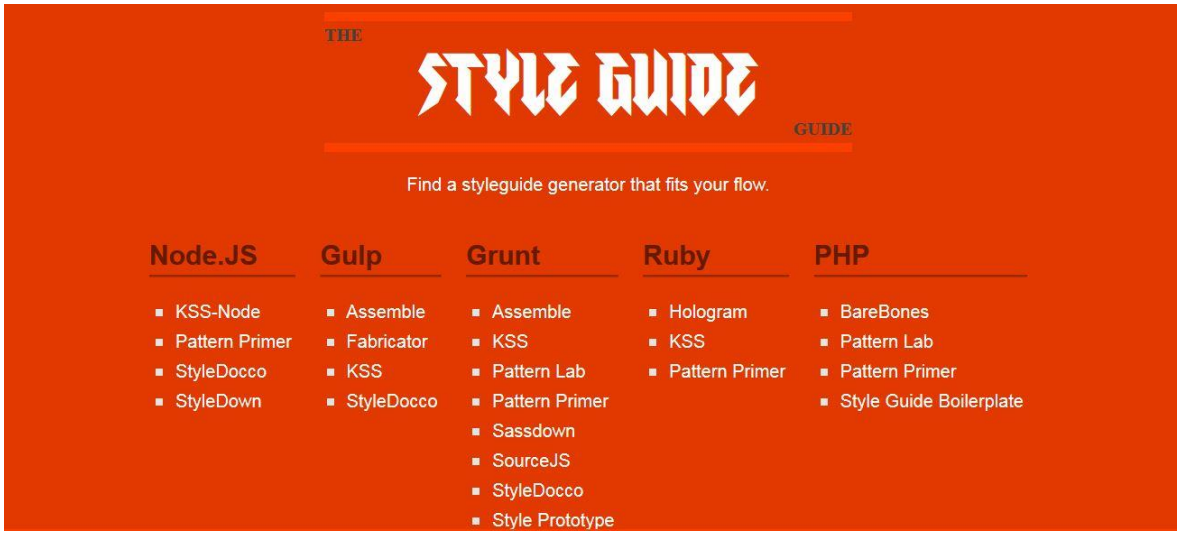
2.6 Tyylioppaan toteuttaminen työkaluilla

Käyttöliittymän tyylioppaan toteutukseen on tehty monia eri työkaluja, jotka toimivat eri ohjelmointikielillä ja automatisointityökaluilla. Automatisointityökalujen tehtävänä on automatisoida useasti toistuvia tehtäviä. Esimerkiksi automatisointityökalu Gruntilla voidaan hoitaa javascriptin minifiointi ja validoiminen sekä css:n minifiointi. Gruntilla myös voidaan tehdä javascript mallipohjat sekä kääntää less-tiedosto css-tiedostoksi. Grunt hoitaa tarvittaessa myös browsersync-toiminnon, jolloin esimerkiksi tyylitiedostoon tehty muutos ajetaan automaattisesti ja browsersync päivittää selaimen ikkunan.

> *“Why use a task runner?*
>
> *In one word: automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it through a Gruntfile, a task runner can do most of that mundane work for you—and your team—with basically zero effort.” (Grunt 2014.)*

2.6.1 Tyylioppaan toteutuksen työkalut

Tyylioppaan toteutukseen on kehitetty monia työkaluja, joita on nähtävillä kuvasta 9 olevasta ruutukaappauksesta The Style Guide Guide -sivulta (The Style Guide Guide 2014.). Sivulle on koottu eri ohjelmointikielillä ja automatisointityökaluilla toimivia tyylioppaan luomistyökaluja, jotka eroavat toisistaan toiminnoillaan ja toiminta-alustaltaan. Esimerkiksi KSS ja StyleDocco muistuttavat paljolti toisiaan, mutta suurimpana erona on se, että StyleDocco:ssa listautuu tyylioppaaseen nähtäville myös komponentin tyylimääreet toisin kuin KSS:ssä. Jotkin tyylioppaat kuten Pattern Lab:lla luodut puolestaan ovat pelkästään komponenttien esittelemiseen keskittyviä, eivätkä välttämättä tuo nähtäville komponentin toteuttamiseen tarvittavaa html-koodia (Pattern Lab demo 2015).



Kuva 9. Kuvassa on kerätty tyylioppaan eri luontityökaluja The style guide guide Internet-sivulle (The Style Guide Guide 2014.)

Edellä esitellyistä työkaluista toimeksiantajan yritys on jo ennestään käyttänyt KSS -työkalua ja Grunt -automatisointityökalua tyylioppaan luomiseen erään asiakkaan projektiin. Tämä vaikutti paljolti käytettävien tekniikoiden valintaan, sillä koettiin helpoimmaksi ja varmimmaksi käyttää samoja tekniikoita, joita yrityksen työntekijät osaavat käyttää jo ennestään. Näin vältyttiin sekä mahdollisilta yhteensopivuusongelmilta tulevaisuudessa että henkilöstön uudelleenkouluttamisen tarpeelta. Tyylioppaan luontityökalun valintaan vaikutti myös se, että toimeksiantajan toiveena oli, että tyylioppaan generoiminen tapahtuisi mieluiten joko Grunt tai Gulp -tehtävänä.

2.6.2 KSS (Knyle Style Sheets)

KSS on laajalti käytetty työkalu tyylioppaan tekoon. Muun muassa Github ja Lonely Planet sekä Veikkaus kertovat käyttävänsä tätä työkalua tyylioppaissaan tai tyylioppaan alareunassa lukee, että tyyliopas on generoitu KSS:ää käyttäen.

KSS on suunniteltu helpottamaan tyylioppaan luomista CSS dokumentointiin perustuvalla tekniikalla, jota kehittäjä pystyy lukemaan helposti tyylitiedostosta. Samalla se on rakenteeltaan sellaista, että myös kone pystyy sitä tulkitsemaan ja tuottamaan tyylioppaan perustuen tyylitiedoston dokumentaatioista. (KSS 2014.)

Tyylitiedoston dokumentointi tapahtuu KSS:ssä seuraavasti. Mikäli käytetään normaalia CSS-tyylitiedostoa, ohjeistuksena on käyttää /* ja */ merkintöjä merkkaamaan dokumentoitua aluetta, kuten CSS-tiedostoissa yleensäkin. Mikäli käytetään esimerkiksi LESS- tai

SCSS –esiprosessoria on suositeltua käyttää // merkintää joka rivin alussa, joka halutaan kommentoida. (KSS syntaksi 2014.)

```
/*
A button suitable for giving stars to someone.

:hover             - Subtle hover highlight.
.stars-given       - A highlight indicating you've already given a star.
.stars-given:hover - Subtle hover highlight on top of stars-given styling.
.disabled          - Dims the button to indicate it cannot be used.

Styleguide 2.1.3.
*/
```

```
// A button suitable for giving stars to someone.
//
// :hover             - Subtle hover highlight.
// .stars-given       - A highlight indicating you've already given a star.
// .stars-given:hover - Subtle hover highlight on top of stars-given styling.
// .disabled          - Dims the button to indicate it cannot be used.
//
// Styleguide 2.1.3.
```

Kuva 10. KSS dokumentaatiomallit (KSS syntaksi 2014.)

Kuvassa 10 on esitelty tyylitiedostoon tehtävät dokumentaatiot, joiden avulla KSS luo tyylioppaan. Kuvassa 10 ylempi dokumentaatiomalli on tarkoitettu käytettäväksi CSS-tiedostossa ja alempi malleista on ohjeistus esimerkiksi LESS-tiedostoon. Kehittäjän on myös helppo lukea dokumentaatiota, koska se on selkeästi tekstimuodossa toteutettu. Samalla dokumentaatio on kuitenkin sellaista, että kone osaa kääntää sen myös tyylioppaaksi. Esimerkeissä on listattu myös elementin pseudoluokat kuten esimerkiksi kuvassa oleva hover-pseudoluokka. Lisäluokat kuten ”disabled” on myös dokumentoitu. Alimpana oleva dokumentaatio ”Styleguide 2.1.3” on osa rakennetta, jonka avulla KSS osaa jäsentää komponentin oikeaan kohtaan hierarkisessa rakenteessa. Kuvassa 10 oleva esimerkki tähtiarvostelun painikkeesta tulisi tyylioppaassa jäsentymään toiseen osioon sen ensimmäisen alaosion kolmanneksi painikkeeksi. Esimerkin tilanteessa 2.0 on esimerkiksi ”Painikkeet”, jonka alle kohtaan 2.1.3 esimerkkinä toimiva tähtien antamiseen tarkoitettu painike tulisi tyylioppaassa sijoittumaan.

KSS on tarkka siitä, että sen vaatimat dokumentaatiot on kirjoitettu oikein ja jopa pieni virhe dokumentaatiossa, kuten välin unohtaminen voi estää tyylioppaan luonnin. Myöskin välilyönneillä ja sisennyksillä on merkitystä lopputuloksen kannalta.

```
// Markup:  <div class="color-box bg-light-grey">@light-grey</div>
// <div class="color-box bg-grey">@grey</div>
// <div class="color-box bg-dark-grey">@dark-grey</div>
// <div class="color-box bg-basic-red">@basic-red</div>
//          <div class="color-box bg-dark-red">@dark-red</div>
//          <div class="color-box bg-brown">@brown</div>
//          <div class="color-box bg-bg">@bg</div>
//          <div class="color-box bg-white">@white</div>
//          <div class="color-box bg-border">@border</div>
//
```

```
<div class="color-box bg-light-grey">@light-grey</div>
<div class="color-box bg-grey">@grey</div>
<div class="color-box bg-dark-grey">@dark-grey</div>
<div class="color-box bg-basic-red">@basic-red</div>
                        <div class="color-box bg-dark-red">@dark-red</div>
                        <div class="color-box bg-brown">@brown</div>
                        <div class="color-box bg-bg">@bg</div>
                        <div class="color-box bg-white">@white</div>
                        <div class="color-box bg-border">@border</div>
```

Kuva 11. Esimerkki KSS dokumentaatiosta

Esimerkkinä kuvassa 11 on esitetty kuinka sisennys vaikuttaa lopputulokseen KSS tyylioppaan merkinnöissä. Kuvasta käy ilmi, että sisennetty dokumentaatio listautuu myös tyylioppaassa sisennettynä. Tämä tulee muistaa tyyliopasta tehdessä, jotta tyylioppaasta saa generoitua halutun näköisen. Mikäli esimerkin kuvasta jättäisi merkinnän ”Markup:” pois, eivät sen alla olevat dokumentaatiot listautuisi näkyviin tyylioppaaseen. Tätä ominaisuutta voidaan hyödyntää esimerkiksi tyylioppaassa ikonien ja värien kohdalla, jolloin ei välttämättä haluta tyylioppaaseen näkyviin muuta kuin pelkät ikonit tai värit. Kuvan 11 esimerkissä dokumentaation ”color-box”-luokka on esimerkiksi tehty vain tyyliopasta varten, jotta tyylioppaaseen saa värit esitettyä neliönmuotoisessa laatikossa. Esimerkiksi tässä tapauksessa ei välttämättä tarvitse markup-tietoja näkyviin, koska tyylioppaaseen generoituu näkyviin kuitenkin jokaiselle värille oma neliö, jonka sisällä kyseinen väri on ,sekä sen muuttuja kuten @white. Tämä toimintatapa selkeyttää tyylioppaan ulkoasua.

3 Tyylioppaan toteutus

Tässä osassa perehdytään tyylioppaan toteutukseen valitulla KSS työkalulla. Tarkoituksena on toteuttaa toimeksiantajalle käyttöönotettava käyttöliittymän tyyliopas heidän Ideapp -sovellukseensa. Toimeksiantaja ei varsinaisesti rajoittanut käytettävää tekniikkaa tai kieltä, mutta toiveena oli, että generointi toimisi Grunt- tai Gulp -tehtävänä. Toiveena oli myös, että tyyliopas olisi selkeä ja helppo käyttää.

KSS valittiin tyylioppaan tekemiseen työkaluksi muun muassa siksi, että se oli jo ennestään toimeksiantajan yrityksessä käytössä, joten se oli tuttu työntekijöille entuudestaan. Näin työntekijöiden ei tarvitse opetella toisen tyylioppaan luomistyökalun merkintöjä ja työskentely pysyy jatkossakin helppona. Ei myöskään nähty syytä vaihtaa toiseen tyylioppaan luomistyökaluun, koska KSS oli todettu toimivaksi yrityksen tarpeisiin. Lisäksi mahdollinen avun tarve tyyliopasta tehdessä vaikutti valintaan, koska näin pystyi tarvittaessa kysymään KSS:ään liittyviä neuvoja yrityksen työntekijöiltä. KSS koettiin muutenkin helpoksi oppia ja käyttää sekä todettiin, että usea yritys käyttää juuri nimen omaa KSS:ää heidän tyylioppaissaan, joten KSS:n osaaminen mahdollisesti tulevaisuutta varten koettiin järkevimmäksi työkaluksi toteuttaa tyyliopas.

3.1 Lähtötilanne työn tekemiseen

Lähtötilanteena työn tekemisessä on se, että sovellusta johon tyyliopasta ollaan tekemässä, on kehitetty jo melko pitkälle ja se on myös asiakaskäytössä muutamilla asiakkailla. Tämä vaikuttaa eniten siihen, että komponentit ja rakenteet ovat jo pitkälti kehitetty valmiiksi eikä tarvitse aloittaa niin sanotusti tyhjältä pöydältä pelkkien ulkoasumallien kanssa. Komponentit eivät ole kuitenkaan tällä hetkellä uudelleenkäytettävissä, vaan ne on toteutettu vain siihen paikkaan, jossa ne tällä hetkellä ovat ja saattavat periä useampiakin eri tyylimääritteitä niitä ympäröivistä rakenteista. Liitteissä 1-3 esitellään Ideapp -sovelluksen ulkoasua ennen tyylioppaan tekoa.

Tyylioppaan tekeminen eroaa tilanteesta, jossa sovellusta lähdettäisiin toteuttamaan niin sanotusti ”tyhjästä” samaan aikaan siinä, että komponentit ja rakenteet voisi suoraan tehdä tyylioppaaseen ja tuoda ne sieltä sovelluksen käyttöön. Koska Ideapp:ia on kuitenkin kehitetty jo melko pitkälle, tulee sovelluksesta erikseen etsiä kaikki siinä käytetyt komponentit sekä rakenteet ja siirtää ne tyylioppaaseen. Samalla niiden toteutus tulee yhdenmukaistaa, jotta saataisiin uudelleenkäytettäviä komponentteja ja rakenteita. Tyylioppaan tekeminen jo olemassa olevaan projektiin, jossa on jo valmiiksi osittain tyylioppaan tekemiseen tarvittavia komponentteja ja tiedostoja, eroaa täysin tyhjästä lähdettäessä

myös siinä, että toimeksiantajan Ideapp -sovelluksen kehityksessä käytetään jo esimerkiksi valmiiksi Grunt:ia, joten siihen tarvitsee vain lisätä tyyliopastehtävä. Muut tehtävät kuten LESS-tiedostojen kääntäminen CSS-tiedostoiksi on jo siis valmiiksi lisätty Grunt:iin.

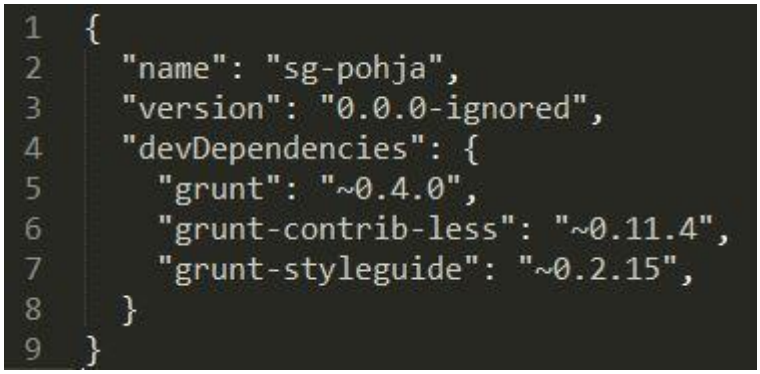
Opinnäytetyön tekemisessä käytettävälle kannettavalle tietokoneelle oli asennettu jo aikaisemmin WampServer, jota käytetään pyörittämään Ideapp -sovellusta paikallisesti koneella sekä Sublime Text 2 -tekstinkäsittelyohjelma LESS- ja HTML- sekä muiden vastaavien tiedostojen käsittelyyn. Myöskin Grunt oli jo asennettuna koneeseen. Versionhallinnasta huolehdittiin käyttämällä SourceTree -nimistä ohjelmaa, joka vie uusimmat versiot sovelluksen ja tyylioppaan tiedostoista BitBucket -nimiseen versionhallintaan. Tyyliopasta lähdettiin tekemään KSS:n perus mallipohjalle, joka on saatavilla muun muassa KSS:n sivuilta (KSS 2014).

WampServer:n käyttö mahdollistaa Ideapp-sovelluksen ajamisen paikallisesti koneella. Sovellukseen pääsee tämän avulla helposti selaimella. Paikallinen serveri vastaa osoitteesta localhost:8888, jolloin päästään käyttämään koneelle ladattua kopiota sovelluksesta. WampServerin configuraatio-tiedostosta täytyy muistaa kuitenkin käydä muuttamassa hakemistopolku vastaamaan oikeata polkua siihen hakemistoon, jossa paikallinen kopio sovelluksesta on. Tyyliopas löytyy myös paikallisesta osoitteesta localhost:8888/hakemistopolku tyylioppaalle. Tässä tapauksessa localhost:8888/docs/styleguide/.

3.2 Grunt

Gruntin asennus koneelle onnistuu gruntjs.com -sivujen ohjeiden mukaan ajamalla komentotulkissa ”npm install -g grunt-cli” komennon, joka asentaa koneelle Gruntin tekstipohjaisen käyttöliittymän. Tämä lisää ”grunt” -komennon PATH-ympäristömuuttujaan ja mahdollistaa komennon ajamisen missä tahansa kansiossa tämän jälkeen. Tämä ei kuitenkaan asenna Gruntin automatisointityökalua vaan mahdollistaa Gruntin ajamisen siinä kansiossa jossa se on. Näin ollen koneella voi olla useampi Grunt asennettuna yhtäaikaisesti. (Grunt asennus 2014.)

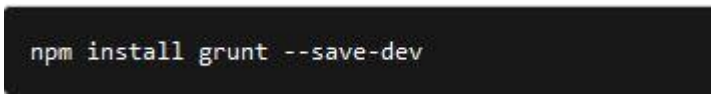
Gruntin tuominen projektiin vaatii kahden tiedoston luontia projektiin, package.json- ja Gruntfile -tiedoston. Package.json -tiedosto sisältää tiedot projektissa käytettävästä Gruntista ja Grunt-lisäosista, kuten esimerkiksi browser-sync:stä. Gruntfile.js -tiedostoa puolestaan käytetään grunt-tehtävien (tasks) konfiguroimiseen tai määrittelyyn sekä Grunt-lisäosien (plugins) lataamiseen.



```
{
  "name": "sg-pohja",
  "version": "0.0.0-ignored",
  "devDependencies": {
    "grunt": "~0.4.0",
    "grunt-contrib-less": "~0.11.4",
    "grunt-styleguide": "~0.2.15",
  }
}
```

Kuva 12. Esimerkki package.json -tiedostosta.

Esimerkkinä kuvassa 12 on kuvankaappaus osasta package.json -tiedostoa, johon on lisätty less- ja styleguide-lisäosat, joilla ajetaan kyseiset grunt-tehtävät. Gruntin ja sen lisäosien asentaminen onnistuu helpoiten jo olemassa olevaan package.json -tiedostoon ajamalla komentotulkissa komennon ”npm install grunt <moduulin nimi> --save-dev”. Komento asentaa halutun lisäosan ja lisää sen samalla devDepencies -osioon package.json -tiedostoon. Kuvassa 13 kuvakaappauksena oleva komento komentotulkissa asentaa Gruntin viimeisimmän version. Se myös samalla lisää sen package.json -tiedostoon devDependencies -osioon.



```
npm install grunt --save-dev
```

Kuva 13. Gruntin asennus.

Tyyliopasta varten Grunt:iin täytyy tehdä muutamia konfigurointeja. Gruntfile -tiedosto sisältää grunt-tehtävät, jotka halutaan ajaa Gruntilla. Esimerkiksi style guide -tehtävä voidaan ajaa määritellyllä komennolla, jolloin Grunt tarkastaa Gruntfile -tiedostosta tehtävää vastaavat toimet ja suorittaa ne.

```
styleguide : {
  kss : {
    options : {
      template : {
        src : 'assets/styleguide-template'
      },
      framework : {
        name : 'kss'
      },
      name : 'Ideapp Styleguide'
    },
    files : {
      'docs/styleguide' : 'css/styleguide.less'
    }
  }
}
```

Kuva 14. Gruntfile.js:än styelguide –tehtävä

Kuvassa 14 olevassa Grunt-tehtävässä on määriteltynä mitä mallipohjaa tyylioppaan tekoon käytetään ja mistä hakemistopolusta mallipohja löytyy. Tämä on kerrottu kohdassa template -> src. ”Framework” kohdassa kerrotaan millä työkalulla tyyliopas tehdään eli tässä tapauksessa KSS. ”Name” kohtaan on määritetty tyylioppaan nimi. ”Files” kohdassa määritellään ensin kohde johon tyyliopas generoidaan (docs/styleguide) ja sen jälkeen lähde (css/styleguide.less), josta generointi tapahtuu.

```
  // These plugins provide necessary tasks.
  grunt.loadNpmTasks('grunt-contrib-clean');
  grunt.loadNpmTasks('grunt-contrib-concat');
  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-qunit');
  grunt.loadNpmTasks('grunt-contrib-less');
  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');
  grunt.loadNpmTasks('grunt-newer');
  grunt.loadNpmTasks('grunt-plato');
  grunt.loadNpmTasks('grunt-contrib-cssmin');
  grunt.loadNpmTasks('grunt-browser-sync');
  grunt.loadNpmTasks('grunt-templates-hogan');
  grunt.loadNpmTasks('grunt-ftp-deploy');
  grunt.loadNpmTasks('grunt-styleguide');
  // Default task.
  grunt.registerTask('default', ['jshint','clean','hogan','less','cssmin','concat']);
  grunt.registerTask('minify', ['jshint','clean','less', 'concat', 'uglify']);
  grunt.registerTask('syncwatch', ['browserSync','watch']);
  grunt.registerTask('sg', ['styleguide']);
};
```

Kuva 15. Grunt-tehtävät on määritelty Gruntfile-tiedostossa

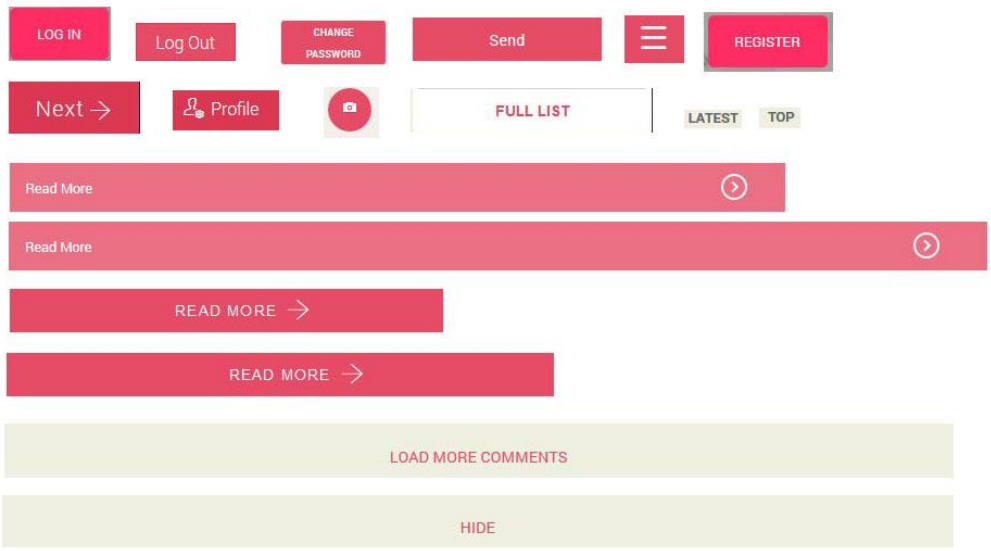
Gruntfile-tiedostoon pitää lisätä vielä kuvan 15 osoittamat määrittelyt esimerkiksi siitä, millä komennolla pystyy jatkossa suorittamaan tyylioppaan generoimisen. Oheisessa kuvassa tyylioppaan generoiminen onnistuu ”grunt sg” -komennolla komentotulkissa.

3.3 Komponenttien erittely ja listaaminen sovelluksesta

Tyyliopasprojektin tekeminen aloitetaan listaamalla tarvittavat tyylioppaaseen tulevat komponentit ja rakenteet sekä muut osat kuten fontit ja värit sovelluksesta. Komponenttien hakemiseen ja etsimiseen sovelluksesta käytettiin esimerkiksi yksinkertaisesti selaimen ”Inspect Element” -toimintoa. Tämän avulla saatiin helposti selville muun muassa minkä nimisiä rakenteita ja komponentteja sovelluksessa on, sekä mistä tyylitiedostosta ne mahdollisesti saavat tyyliinsä vaikutteita.

Tyylioppaaseen tulevia komponentteja:

- painikkeet
- värit
- typografia
- listat
- otsake
- moduulit (esimerkiksi arvostelupalkki sovelluksen alareunassa.)
- input-kentät
- sovelluksessa olevat ”palkit”.



Kuva 16. Ideapp:sta kerätyt painikkeet

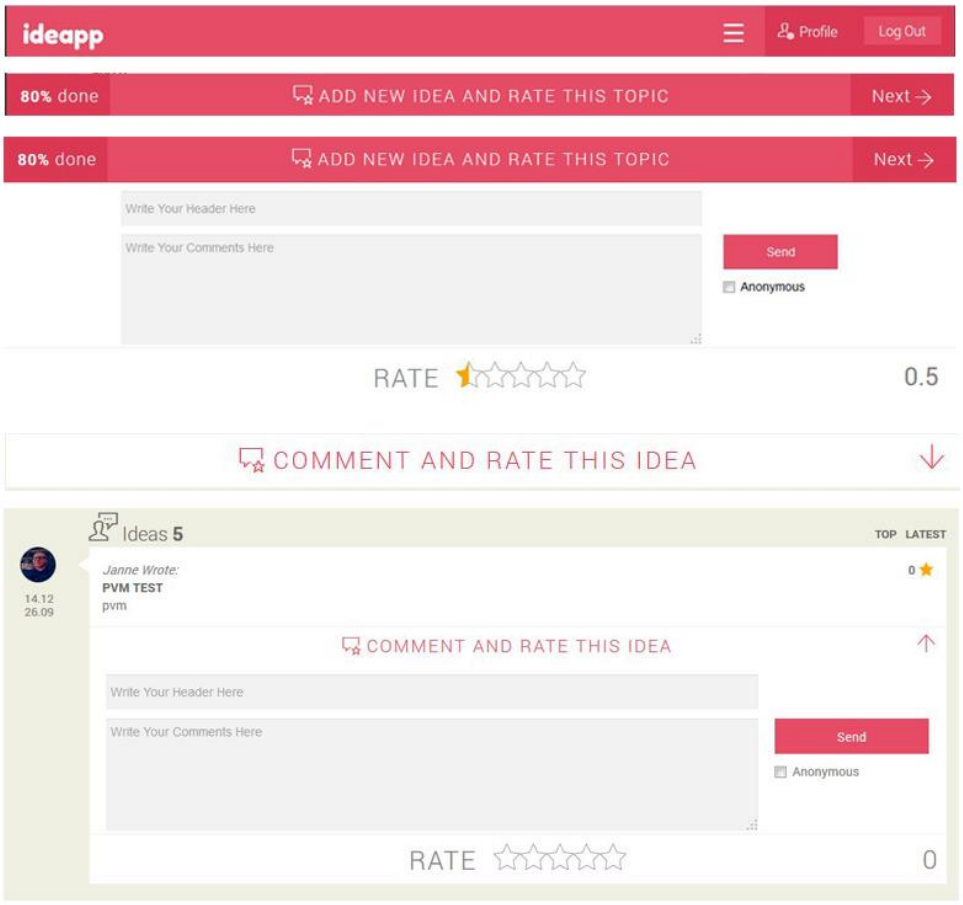
Oheisessa kuvassa 16 on koottuna Ideapp:sta tällä hetkellä löytyviä painikkeita. Kuvasta näkyy se, että Ideapissa käytetään tällä hetkellä muutamaa erinäköistä painiketta useassa eri kohdassa sovellusta. Ongelma onkin siinä, että vaikka painikkeet saattavat näyttää samanlaisilta, ne ovat kuitenkin luotu vain siihen kohtaan sovellusta jossa ne ovat. Tällainen kehittämistyyli johtaa siihen, että niitä ei pystytä suoraan käyttämään toisessa kohdassa sovellusta. Tämä johtuu siitä, että ne perivät määritteitä niistä rakenteista joiden sisällä ne ovat.

Kuva 17. Ideapp:ssa käytettäviä värejä

Oheisessa kuvassa 17 on kerätty Ideapp:ssa tällä hetkellä käytössä olevia värejä lukuun ottamatta täysin mustaa ja valkoista. Sovelluksessa käytetyistä väreistä harmaan ja ruskean sävyjä käytetään varsinkin teksteissä ja punaisen sävyjä muun muassa otsakkeessa ja sovelluksessa käytettävissä rakenteissa. Vaalean harmaita sävyjä käytetään taustavärein sovelluksen eri elementeissä. Sovelluksessa käytetään melko montaa väriä tällä hetkellä ja esimerkiksi teksteissä värisävyjä käytetään erottamaan niiden käyttötarkoitusta. Esimerkiksi eri värisävyillä erotetaan niin sanottu ”leipäteksti” vaikkapa otsikosta. Joitain läheisiä värisävyjä saatetaan tulla poistamaan yhtenäisemmän ulkoasun ja helpomman ylläpidettävyyden takia. Tällaisia sävyjä löytyy muun muassa käytettävissä olevista harmaan sävyistä.

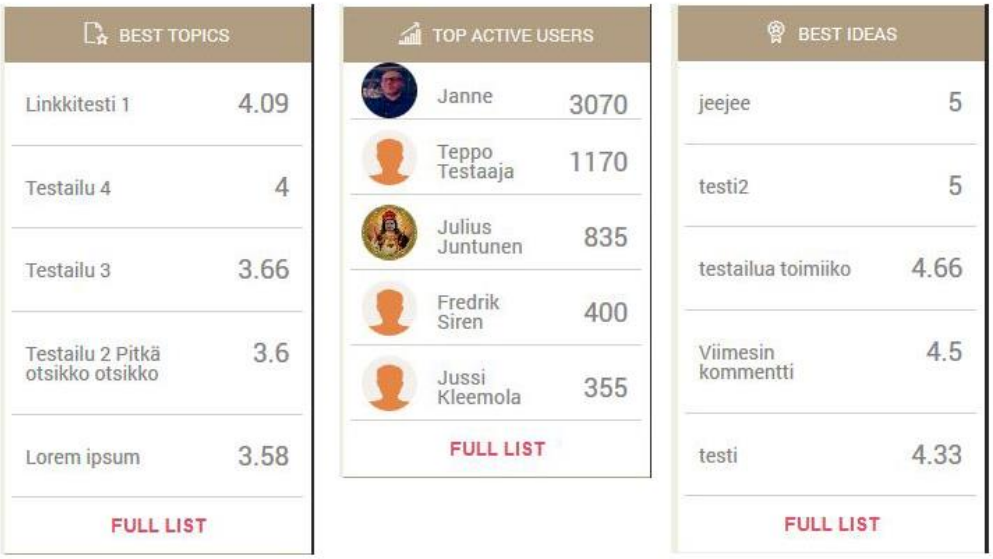
Kuva 18. Esimerkki kolmesta löydetystä erilaisesta H4 typografiasta

Kuvassa 18 on kerätty kolme erilaista otsikkoa, joissa kaikissa on käytetty H4-tason otsikkoa, mutta ne ovat kaikki erinäköisiä keskenään. Yhteensä H4-otsikoita löytyy sovelluksesta ainakin viisi erilaista eri tilanteissa käytettynä. Ideana olisi yhdenmukaistaa niin, että käytössä olisi vain yksi H4-otsikko, joka näyttää samanlaiselta riippumatta missä se on käytössä. Tarkoituksena on siis luoda vain selkeät ylimmän tason otsikot eli h1, h2 ja niin edespäin, jotta saadaan selkeämpi rakenne sovellukseen.



Kuva 19. Ideapp:n rakenteita

Kuvassa 19 on koottu Ideapp:sta löytyviä rakenteita kuten otsake ja muutama ”palkki”, joista punainen ”add new idea and rate this topic” -palkki on toteutettu sivun alareunaan. Palkki avautuu kuvassa näkyväksi kokonaisuudeksi, jolla pystyy arvostelemaan sen hetkistä avoimena olevaa keskusteluaihetta. ”Comment and rate this idea” -rakenne puolestaan on jokaisen aihetta ideoineen ja kommentoineen kommentin perässä, jotta muut henkilöt pystyvät vielä kyseistä ideaa arvostelemaan ja kommentoimaan.



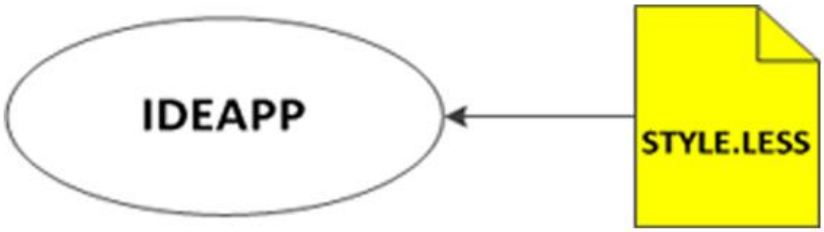
Kuva 20. Lista-elementtejä kerättynä Ideapp:sta

Oheisessa kuvassa 20 on kerätty lista-elementtejä Ideapp:n ”sivupalkista”. Vaikka listat näyttävät melko samoilta lukuun ottamatta tekstejä, löytyy niidenkin toteutuksesta hieman eroavaisuuksia. ”Top active users” -lista eroaa kahdesta muusta listasta esimerkiksi sillä, että siinä on erilaiset tyylimääritteet, koska se sisältää avatar-kuvan.

3.4 Tyylioppaan toteuttaminen

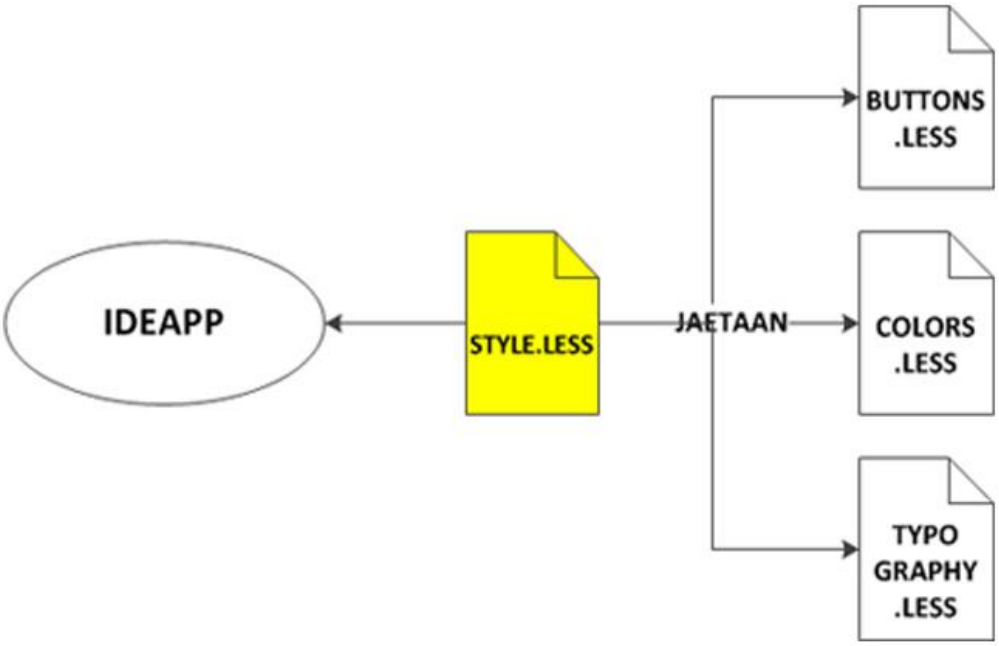
Varsinainen idea työssä on se, että style.less -tiedostoon on tällä hetkellä tehty kaikki Ideapp:n elementit, josta ne tulee etsiä erikseen ja jakaa selkeyttäviin omiin less-tiedostoihin kuten painikkeet esimerkiksi buttons.less -tiedostoon. Tämän jälkeen uudet less-tiedostot tuodaan style.less-tiedostoon käyttöön. ”Importtaaminen” eli tiedoston tuominen onnistuu yksinkertaisesti lisäämällä style-tiedoston alkuun esimerkiksi ”@import ”buttons.less”;”. Näin menetellään kaikkien tiedostojen kanssa, jotta saadaan luotua niiden välille yhteys ja mahdollistettua toisessa tiedostossa olevan datan käyttö style.less-tiedostossa.

Komponenttien ja rakenteiden muokkaamisessa ja toteutuksessa tullaan ottamaan huomioon se, että sovellus on suunniteltu tällä hetkellä toimivaksi vain Ipad- ja tietokoneversioina, joten esimerkiksi mobiilin mahdollisesti vaativia muutoksia komponentteihin ei tulla ottamaan huomioon. Mobiilista ei ole tällä hetkellä ulkoasusuunnitelmia, joten siinä vaiheessa kun yritys päättää laajentaa sovelluksen toimintaa myös mobiiliystävällisemmäksi, komponentteihin ja rakenteisiin voi joutua tekemään tarvittavia muutoksia yhteensopivuuden takaamiseksi.



Kuva 21. Lähtötilanne

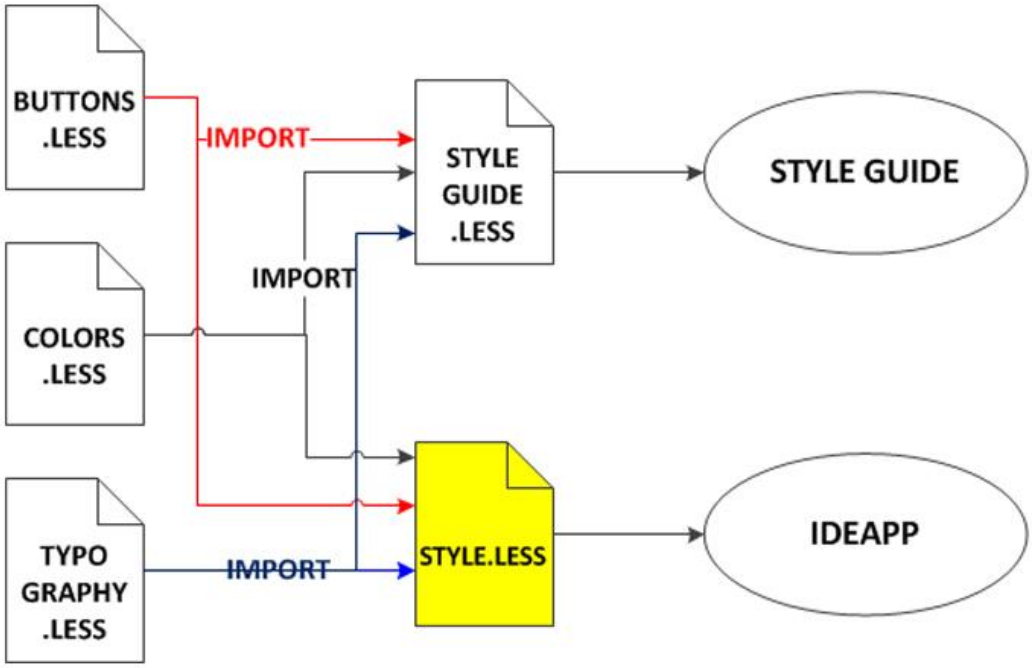
Kuvassa 21 on kuvattuna se lähtötilanne, josta työtä lähdetään tekemään. Style.less -tiedosto on Ideapp:n tyylitiedosto, johon on sisällytetty kaikki tyylit joita Ideapissa käytetään. Sen koko ennen tyylioppaan tekemisen aloitusta on noin 53KB:tä ja siinä on 2670 riviä koodia. Tarkoituksena on etsiä tuosta tiedostosta erilleen kaikki sovelluksen komponentit ja ryhmitellä ne omiin ryhmiinsä pienenpiin tyylitiedostoihin.



Kuva 22. Komponenttien erotteleminen omiin tiedostoihin

Kuva 22 kuvaa työn vaihetta, jossa style.less -tiedostosta etsitään komponentit ja muut halutut asiat kuten värit sekä typografiat ja siirretään ne omiin tiedostoihinsa. Kuvassa on esitetty esimerkkinä kolme uutta tyylitiedostoa, mutta todellisuudessa niitä tulee olemaan

enemmän riippuen mitä kaikkea saadaan eroteltua omiin tiedostoihin. Tämä selkeyttää hakemiston rakennetta ja auttaa löytämään nopeammin halutun tiedon tiedostoista.



Kuva 23. Lopputilanne

Kuvassa 23 on puolestaan esitetty lopputilanne, jossa uudet tehdyt tyylitiedostot sisällytetään takaisin style.less -tiedoston käyttöön niin sanotusti ”importtaamalla” ne tiedostoon. Tämä onnistuu lisäämällä style.less -tiedoston alkuun koodimerkinnän @import ”buttons.less”;. Näin esimerkiksi button.less -tiedoston sisältö saadaan käyttöön myös style.less -tiedostossa, josta se käytetään Ideapp -sovelluksessa. Samalla uudet tyylitiedostot tuodaan myös styleguide.less -tiedostoon, joka on tyylioppaan oma tyylitiedosto. Kun toimitaan näin, voidaan olla varmoja siitä, että tyylioppaassa ja sovelluksessa on käytössä samat komponentit kummassakin.

Uusi järjestely aiheutti muutamia muutoksia tiedostopolkuihin. Tiedostopolkua ikoneille piti muuttaa siirtämällä icons -hakemisto ulos assets -hakemistosta, jotta tyylioppaan ja Ideapp:n tiedostopolut olisivat samanlaiset ja molemmat pystyisivät käyttämään ikoneita. Ikoneille tehtiin oman less-tiedosto, johon siirrettiin kaikki sovelluksen ikonit, jotta ylläpito olisi helpompaa nyt kun kaikki ikonit ovat keskitetty samaan hakemistoon. Edellä mainittu hakemistopolun muutos oli ainoa, joka tarvitsi tehdä, sillä muuten hakemistorakenne oli molemmille sekä tyylioppaalle että sovellukselle samanlainen. Tämän jälkeen tarvitsi luoda vain uusia less-tiedostoja niitä komponentteja ja rakenteita varten, jotka ”kaivettiin” sovelluksesta ulos ja sisällytettiin tyylioppaaseen.

3.5 Värit

Värien nimeäminen on yksi asia johon tulee kiinnittää huomiota. Sovelluksesta haettiin kaikki siinä käytettävät värit ja sisällytettiin ne yhteen colors.less -tiedostoon, josta niitä käytetään. Tällöin huomattiin esimerkiksi se, että sovelluksessa oli nimetty värejä kuten bg ja border, jotka eivät kertoneet sinällään hirveästi väristä vaan lähinnä niistä pystyi arvaamaan missä kyseistä väriä on sovelluksessa käytetty. Toimeksiantajan kanssa juteltua tultiin siihen tulokseen, että colors.less -tiedostossa olevat värit nimetään esimerkiksi @color-light-grey -muuttujilla, jolloin heidän mukaansa värin löytää helpommin etsittäessä sitä tekstieditorissa, kun siinä on tuo color -etuliite. Samoin päätettiin, että värien luokkien nimet tulee alkaa bg, jotta tiedetään, että väri vaikuttaa taustaväriin. Esimerkiksi bg-light-grey -luokka vaikuttaa sen elementin taustaväriin, jolle kyseinen luokka on annettu.

```
1  // Background colors
2  //
3  // Basic background colors used in Ideapp <br>
4  // <span class="text-black">@color-white</span> is used for sidebar <br>
5  // <span class="text-black">@color-light-grey1</span> is used for app background color <br>
6  // <span class="text-black">@color-light-black</span> is used for body background color <br>
7  //
8  // <div class="color-box bg-white"><span>@color-white</span></div>
9  // <div class="color-box bg-light-grey1"><span>@color-light-grey1</span></div>
10 // <div class="color-box bg-light-black"><span class="dif">@color-light-black</span></div>
11 //
12 // Styleguide 3.1
13
14 @color-white : #fff;
15 @color-light-grey1 : #eff0e2;
16 @color-light-black:#282828;
17
18 .bg-white{background-color: @color-white;}
19 .bg-light-grey1{background-color: @color-light-grey1;}
20 .bg-light-black{background-color: @color-light-black;}
21
```

Kuva 24. Kuvankaappaus colors.less -tiedostosta

Tyylioppaan mukana siirryttiin käyttämään myös värimuuttujia kuten @color-white, jotka ovat colors.less -tiedostossa. Kuvassa 24 on nähtävillä muuttujien lisäksi värien luokat sekä tyylioppaan tarvitsemat merkinnät. Värimuuttujien käyttö helpottaa jatkossa sovelluksen kehitystä, koska halutun värin löytää helposti tyylioppaasta ja sen saa käyttöön esimerkiksi @color-white -merkinnällä. Tämä toimintatapa myös helpottaa tyylitiedoston ja sovelluksen ylläpidettävyyttä. Esimerkiksi jos harmaan sävyä tulevaisuudessa halutaan muuttaa vaikka astetta tummempaan, onnistuu muutos helposti muuttamalla colors-less -tiedostoon halutun tulevan sävyn hexadesimaaliarvo vanhan arvon tilalle. Näin muutos vaikuttaa saman tien kaikkiin elementteihin, joilla on annettu värimääritteen arvoksi @color-grey (color: @color-grey;). Aikaisemmin tyylitiedossa oli annettu jokaiseen värimääritteeseen suoraan hexadesimaaliarvo esimerkiksi valkoinen (color: #ffffff;). Uusi toimintatapa helpottaa myös tyylitiedoston luettavuutta, koska uudesta värimuuttujasta on

helpompi ymmärtää mikä väri kyseisessä kohdassa on käytössä, kuin aikaisemmasta tavasta merkitä hexadesimaaliarvo suoraan värin arvoksi.

Värejä päätettiin myös yhtenäistää niiltä osin kuin sen katsottiin olevan tarpeellista ja perusteltua. Sovelluksessa oli esimerkiksi useampi harmaan sävy, jotka erosivat toisistaan vain hieman eikä niiden välillä ollut havaittavissa suurtakaan eroa vierekkäin asetettuna, joten joitakin sävyjä päätettiin jättää pois. Valintaan vaikutti myös se, että näistä muutamaa sävyä käytettiin vain parissa kohdassa muutamaan kertaan. Vähinten käytettyjen sävyjen komponentit vaihdettiin käyttämään toista harmaan sävyä, jota sovelluksessa käytetään enemmän. Samoin tehtiin myös muidenkin värien kohdalla kuten esimerkiksi ruskean. Tämä toi samalla ulkoasuun myös hieman enemmän yhtenäisyyttä, koska värimaailma yhtenäistyi. Projektin alussa etsityistä väreistä osa oli myös jäänyt tyylitiedostoihin vanhasta sovelluksen versiosta eikä niitä enää käytetty missään. Nämä värit päätettiin myös poistaa viemästä niin sanotusti ”turhaa tilaa” tiedostosta.

3.6 Painikkeet

Painikkeiden kanssa oli muutama vaihtoehto niiden toteutuksesta tyylioppaaseen. Vaihtoehtoina oli luoda kokonaan uusi painike tai muokata vanhasta. Painikkeita varten luotiin yksi ”perus”-painike, joka sisältää painikkeiden perustyylit, kuten taustavärin ja painikkeessa käytettävän tekstin oikean fontin. Painikkeet luotiin tämän ”perusmallin” pohjalta muuttamalla haluttuja arvoja, jotta saatiin luotua halutun näköinen painike. ”Perusmallin” etuna on se, että näin saadaan kaikkiin painikkeisiin helposti jaettua samat halutut määritteet ja niitä tarvitsee tulevaisuudessa muokata vain yhdestä kohdasta jolloin vaikutus koskee kaikkia painikkeita. Esimerkiksi jos tulevaisuudessa painikkeiden fonttia haluttaisiin vaihtaa, se onnistuisi helposti muokkaamalla vain perusmallin määrittelyä.

Koska uudet uudelleenkäytettävät painikkeet eivät sisällä valmiiksi muun tiedon painikkeesta kuin sen ulkoasun määritteet, jouduttiin luomaan näiden painikkeiden ympärille uusi ympäröivä elementti, jotta ne saatiin sovelluksessa sijoitettua haluttuun kohtaan. Tämä johtui esimerkiksi otsakkeessa kirjautumis - ja profiili-painikkeiden kanssa siitä, että aikaisemmin molemmat painikkeet oli luotu vain siihen kohtaa sovellusta ja ne sisälsivät myös määritteitä, jotka vaikuttivat niiden sijoitukseen sovelluksessa. Molempien painikkeiden aikaisemmat tyylimääritteet siis sisälsivät sekä painikkeen ulkoasuun, kuten leveyteen ja korkeuteen liittyvät määritteet, että näiden paikkamääritteet otsakkeessa. Uudelle ympäröivälle elementille annettiin sopivasti esimerkiksi margin-left ja margin-top -arvoja, jotta kirjautumis- ja profiili-painikkeet saatiin siirrettyä niille tarkoitetulle paikalle sovelluksessa. Uudessa toimintamallissa on se etu, että painikkeille ei anneta erikseen paikkamääritteitä

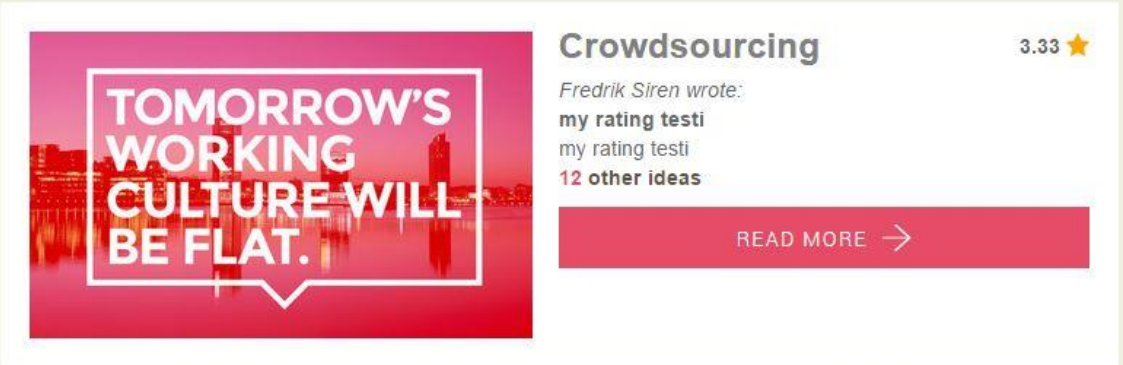
vaan molemmat painikkeet ”asettuvat” omalle paikalleen kun ne ovat saman isäntäelementin sisällä, jolla on annettu paikkamääritteet.

3.6.1 Open topic -painike

Kuvassa 25 on esitetty ”presentaatiolaatikko” ennen mitään muutoksia. Kuvasta nähdään, että ”Read more” -painike on samalla tasolla kuin presentaation kuva eli sillä on tyylimäärittelyissä määreet ”position:absolute” ja ”bottom:0”. Koska ideana on tehdä yleiskäytettävä painike, ei voida laittaa painikkeeseen edellä määriteltyjä tyylimääreitä, sillä muuten painike olisi joka kohdassa tulevaisuudessa aina sitä ympäröivän rakenteen pohjalla.

Kuva 25. Presentaatiolaatikko ennen muutoksia

Kun openTopic-painikkeesta oli tehty oma versio, kokeiltiin sitä Ideapissa jotta nähtäisiin kuinka se toimisi siellä. Tämä onnistui vaihtamalla vanha painikkeen koodi uuteen presentations.hogan-tiedostosta, johon on rakennettu Ideapissa oleva ”presentaatiolaatikko”. Tämä laatikko renderöidään sovellukseen automaattisesti niin monta kertaa kuin on presentaatioita. Esimerkiksi jos sovellukseen on tehty viisi eri presentaatiota, tulostuu sovellukseen viisi kertaa presentaatiolaatikko, jossa jokaisessa laatikossa on juuri siihen presentaatioon liittyvät tiedot näkyvillä. Esimerkiksi presentaation kuva vaihtelee riippuen siitä millaisen kuvan käyttäjä on laittanut.

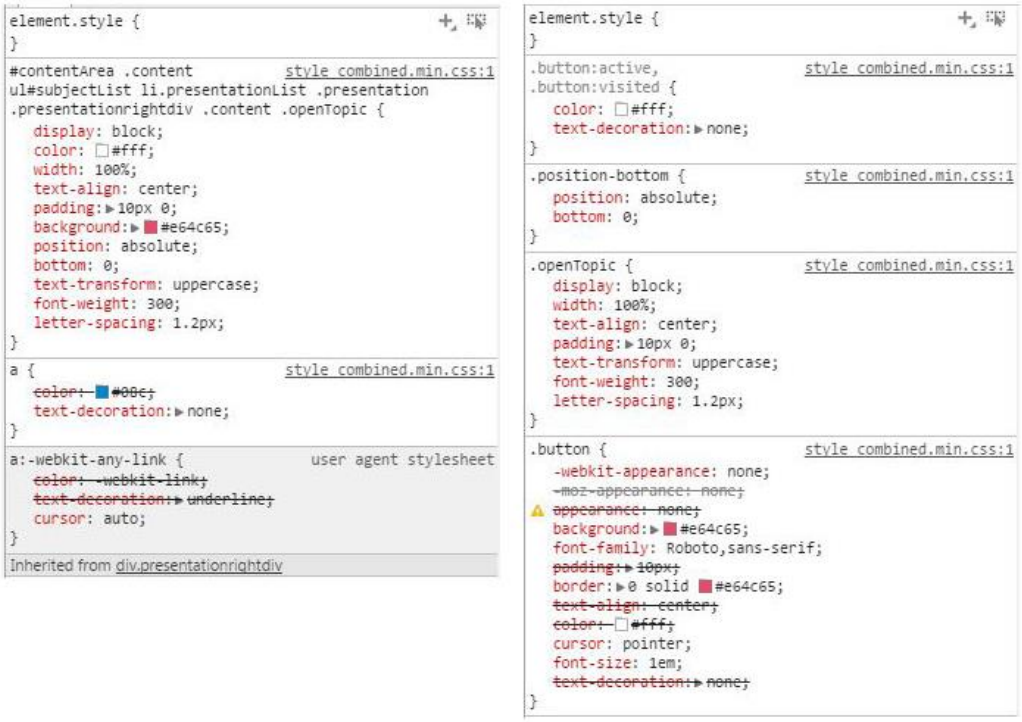


Kuva 26. openTopic -painikkeen kokeilu

Kuvassa 26 on kuvattu tilanne, jossa kokeiltiin ensimmäisen kerran uutta openTopic -painike komponenttia, joka ensin on eritelty sovelluksesta ja muokattu yleispäteväksi komponentiksi, joka pitäisi pystyä toteuttamaan jokaiseen sovelluksen kohtaan samannäköisenä. Kuten kuvasta huomataan, painike ei ole samalla tasolla viereisen kuvan kanssa kuten aikaisemmin. Tämä johtuu siitä, että ennen komponentilla oli määriteltynä sen positioksi absolute ja sillä oli myös määrittely bottom:0, joka asetti painikkeen sitä ympäröivän rakenteen pohjalle. Jotta saataisiin uusi komponentti sitä ympäröivän elementin pohjalle, luotiin uusi luokka, jolle annetaan määrittelyiksi position:absolute ja bottom:0. Tämä luokka annettiin painikkeelle, jotta saavutettu lopputulos saatiin mahdolliseksi.

Kuva 27. Saavutettu lopputulos

Kuvassa 27 on saavutettu lopputulos openTopic -painikkeesta, jolle on annettu myös luokaksi position-bottom. Tällä luokalla saadaan elementille annettua lisämäärittelyitä, joita ei siten tarvitse antaa itse painikkeelle kuten ennen.



Kuva 28. openTopic -painikkeen tyylimääritteet ennen ja jälkeen

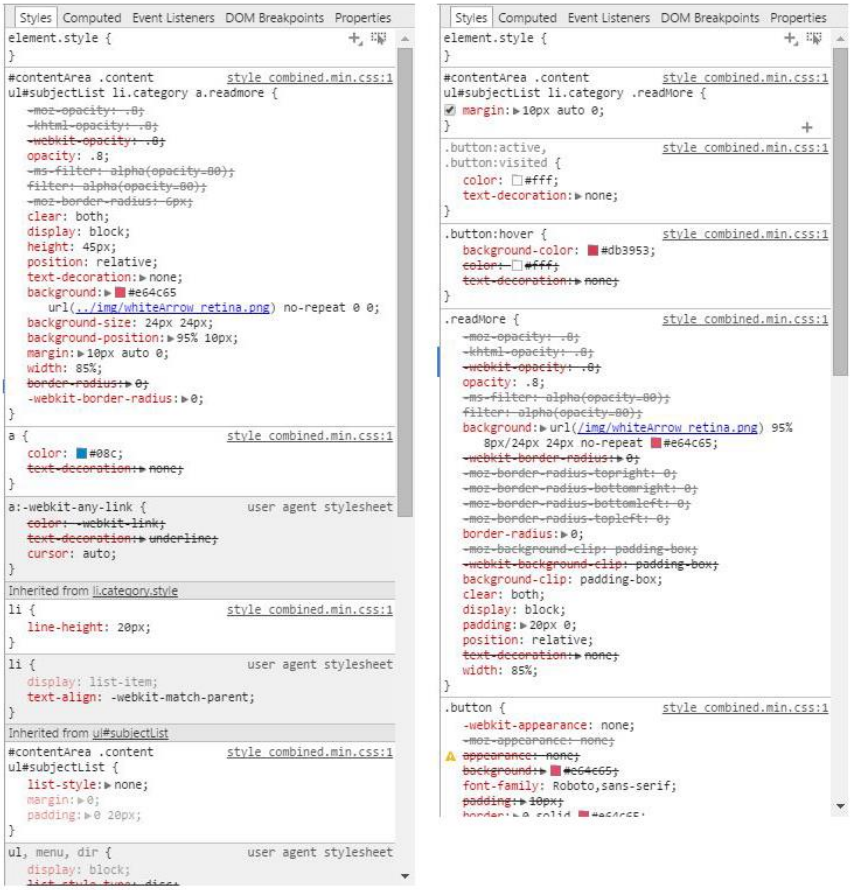
Kuvasta 28 on huomioitavaa myös se, että uusi openTopic-painike saa tyylimäärittelyitä myös luokalta button, joka toimii jokaisen uuden painikkeen pohjana. Button-luokka toimii siis jokaisen uuden painikkeen perustyylinä, jonka pohjalta muokataan uusia painikkeita kuten esimerkiksi edellä mainittu openTopic-painike. Kuvassa vasemmalla olevat määritteet ovat siis vanhat tyylimääritteet ja oikealla olevat ovat uudet.

3.6.2 Read more -painike

Kuvassa 29 on esitelty readmore-painike, jota käytetään Ideapissa kategorianäkymässä jokaisessa kategorialaatikossa, joiden toimintaperiaate on sama kuin edellä mainitun presentaatiolaatikon.

Kuva 29. "Read more"-painike

Ennen painikkeen irrottamista sovelluksesta ja uudelleen muokkaamista se oli koodattu syvälle rakenteeseen kuten kuva 30 osoittaa. Kuvassa 30 on myös nähtävillä uuden painikkeen määrittelyt ja voidaan huomata myös se, että se on ylimmällä tasolla hierarkiassa toisin kuin ennen. Kuvassa vasemmalla olevat määrittelyt ovat vanhat määrittelyt, joita lähdettiin muokkaamaan. Kuvassa oikealla puolestaan ovat uudet määrittelyt. Huomioitavaa on se, että uusissa määrittelyissä readMore -painike saa määrittelyitä button-luokalta. Painikkeen oman ”readMore” -luokan määrittelyt sisältävät vain ne perus ulkoasumääritteet, jotka tekevät painikkeesta halutun näköisen. Muut kuten sijoitteluun vaikuttavat margin tai vastaavat määrittelyt tehdään Ideapp:in käyttämään style.less -tiedostoon, jotta painike saadaan sovelluksessa halutulle paikalle.



Kuva 30. readMore -painikkeen tyylimäärittelyt

Kuvassa 30 oikealla on uuden painikkeen tyylimääritteet ja voidaan huomata, että painike saa paikkamäärittelynsä margin-arvolla eri paikasta kuin ulkoasumäärittelynsä. Paikkamäärittely on annettu sille style.less -tiedostossa, jota sovellus käyttää tyylitiedostonaan. Painikkeen ulkoasuun liittyvät määrittelyt tulevat puolestaan buttons.less -tiedostosta, johon painike on luotu. Luokka readMore sisältää siis painikkeen ulkoasua koskevat tyyli-

määritteet. Painikkeen luokan readMore paikkamääreet kuten ”position:relative;” ovat painikkeen ympyränmuotoisen kuvan paikkamääreet, jotta se saadaan oikealle paikalleen.

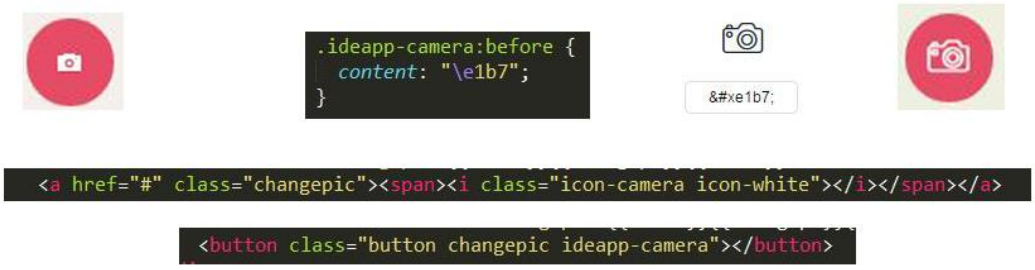
3.6.3 Form -painike

Sovelluksessa on rekisteröintiin ja sisäänkirjaukseen tehty samanlaiset painikkeet, mutta ne ovat kuitenkin tehty erikseen ja saavat tyylimäärittelynsä omien yksilöllisten tunnustensa (ID) mukaan. Koska painikkeet ovat kuitenkin ulkoisesti samanlaiset, tultiin siihen tulokseen, että luodaan niiden pohjalta yksi painike formButton, jota käytetään molemmissa tapauksissa. Sisäänkirjautumisessa painikkeen sisään haetaan tekstitiedostosta oikea teksti ”Sign in” ja rekisteröinnissä puolestaan ”Register”. Näin ollen uudet painikkeet käyttävät samaa formButton -luokkaa, koska ovat ulkoisesti samanlaisia, mutta niille saadaan haettua eri teksti sisällöksi.

Huomioitavaa vanhojen painikkeiden korvaamisessa on tarpeen mukaan säilyttää niiden yksilölliset tunnuksensa. Esimerkiksi ”Loginbutton”- id täytyy säilyttää sisäänkirjautumispainikkeessa, koska sen perusteella painikkeelle linkitetään loginfunktio, joka ajaa sisäänkirjaus-funtion painiketta painettaessa. Sama pätee myös register-painikkeessa. Rekisteröinti painikkeeseen tulee jättää sen yksilöllinen tunnus ”addUser”, jotta rekisteröinti funktio tulee ajetuksi painiketta painettaessa. Edellä mainituilla yksilöllisillä tunnuksilla oli myös omia tyylimääritteitä style.less -tiedostossa, jonne siis on vanhan toimintamallin mukaan toimittuna tehty sovelluksen elementtien tyylimääritteet. Style.less -tiedostosta poistettiin tyylimääritteet, koska jatkossa ulkoasutyylimääritteet tulevat painikkeiden omasta tyylitiedostosta. Style.less -tiedostosta tulee poistaa vanhat tyylimääritteet, jotta ne eivät aiheuta tyylissä sekaannuksia.

3.6.4 Salasananvaihto -painike

Hankalin painike tai ainakin työläin oli käyttäjäkuvan vaihtoon tarkoitettu pyöreä painike. Painike oli alun perin rakennettu a -elementistä, jonka sisällä oli span -elementti. Tälle elementille oli annettu kaikki painikkeen ulkoasua koskevat tyylimääritteet. Span -elementin sisälle oli vielä tehty i-elementti, johon oli laitettu taustakuvaksi kamera. Kamera kuvan kanssa oli aluksi hankaluuksia, koska alun perin kamera oli taustakuvana, joka tuli tyylitiedoston kautta luokkana. Ongelmia oli saada se näkymään myös tyylioppaassa, koska taustakuva haettiin käyttöön vanhemman käytössä olevan Bootstrap-version ikoneista. Ratkaisuksi löytyi se, että yksinkertaistettiin painikkeen rakennetta ottamalla span- ja i-elementit pois ja sisällyttämällä uusi kamera suoraan button-elementtiin. Span -elementiltä poistettiin painikkeen ulkoasua koskevat tyylimääritteet, jotka siirrettiin changepic -luokalle.



Kuva 31. käyttäjäkuvan vaihtopainike

Koska vanha kamerakuva tuli taustakuvana, päätettiin se vaihtaa ikoniksi, sillä sovelluksessa käytetään muuallakin ikoneita ja ne ovat helppoja käsitellä. Kamera-ikonin käyttöönotto oli helppoa, koska sovelluksessa jo käytössä olevasta Ikonifontti-tiedostosta löytyi myös kamera. Käyttöönottoa varten icons.less -tiedostoon lisättiin uusi luokka ja pseudo -elementti ideapp-camera:before, jonka sisällöksi (content) laitettiin kamera-ikonin määritteen neljä viimeistä arvoa, jotta kameran saa näkyviin. Kamera ikonin määrite on ilmoitettu hex-arvolla. Kuten kuvassa 31 nähdään, on kamera-ikonin hex-arvo &#xe17b. Jotta tyylitiedosto osaa tulkita ja käyttää oikeaa ikonia, tulee käyttää sen neljää viimeistä arvoa (e17b). Näin tyylitiedosto osaa ”lukea” koodin ja löytää ikonifontti-tiedostosta oikean ikonin muiden joukosta.

3.7 Typografia

Ideappiin tutustumisen jälkeen tultiin siihen tulokseen, että sen otsikoita tulee selkeyttää ja tehdä vain päätason otsikoita kuten h1, h2 ja h3. Ideapissa oli alun perin erilaisia otsikoita, joista osa, kuten keskustelunaiheen otsikko ”topicHeader”, oli luotu span -elementin avulla. Tämä sai tyylinsä sille annetulta topicHeader-luokalta. Päätason otsikoiden käyttö selkeyttää sovellusta ja sen ylläpitoa, koska samaa otsikkoa pystyy tulevaisuudessa käyttää haluamassaan kohdassa antamalla yksinkertaisesti valitsemalla halutun otsikon tason.

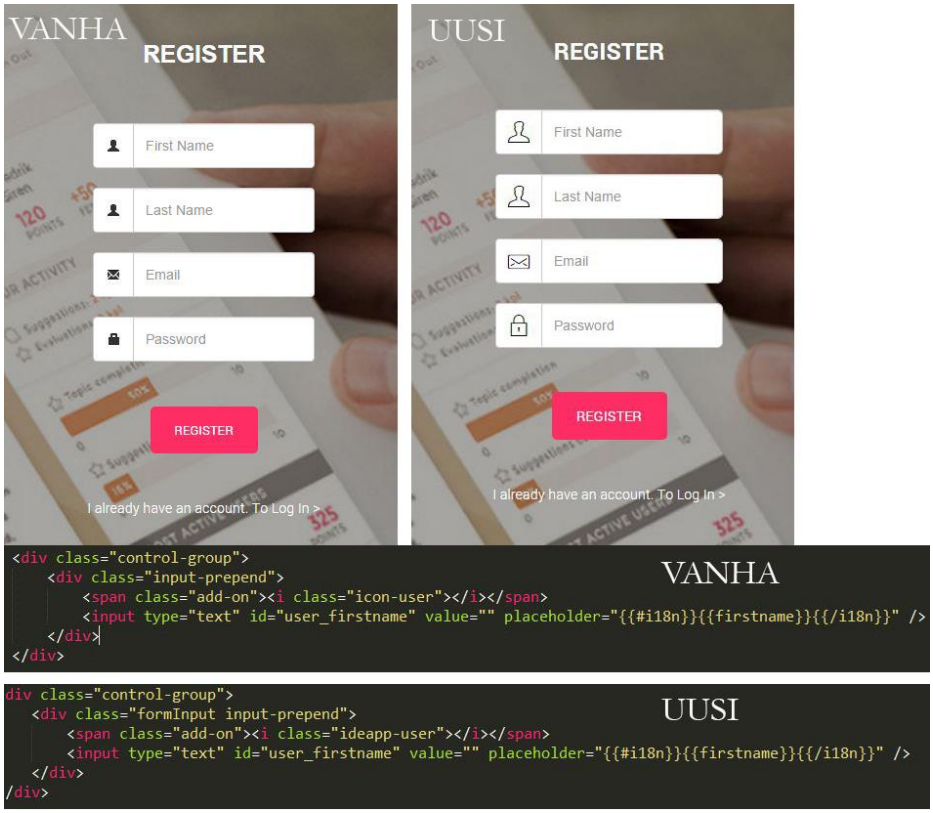
Kuva 32. Esimerkki muutamasta otsikosta Ideapp:ssa

Kuvassa 32 on esimerkkinä muutama otsikko Ideapp:sta. Yhtenäisemmän ulkoasun saamiseksi päätettiin h4-otsikko ja topicHeader yhdistää h1-tason otsikoksi, koska ne olivat kirjasinkooltaan isoimmat otsikot Ideapp:ssa. Niiden yhdistäminen tuntui myös luontevalta sen takia, että se yhtenäisti sovelluksen yleistä olemusta. Molemmat otsikot ovat nimittäin kytköksissä toisiinsa sillä molempiin ”renderöidään” eli ruudulle piirretään sama sisältö. H4-tason otsikko on käytössä niin sanotussa ”topic-näkymässä”, josta siirrytään itse ”topikkiin” eli aiheeseen, jonka otsikkona topicHeader on. Tämän takia oli myös luonnollista yhdistää niiden ulkoasua.

3.8 Lomakkeet ja syöttökentät

Ideapp:n syöttökentät oli rakennettu käyttäen Bootstrapin tyylimäärittelyitä. Ideapp:ssa oli käytössä vanhan 2.3.3 version Bootstrapin grid- ja form -elementit tavallisen css-tyylitiedoston kautta. Jotta tyylioppaaseen saatiin käyttöön samat määritteet syöttökentille Bootstrapista, poistettiin turhat Bootsrapin tyylitiedostot ja tuotiin sovelluksen ja tyylioppaan käyttöön vain tarvittavat less-tyylitiedostot Bootstrapista, kuten forms.less ja grid.less -tiedostot. Näin saatiin pois samalla turhat Bootstrapin tuomat määrittelyt, jotka muun muassa sekoittivat joitain typografioita. Toinen vaihtoehto olisi ollut muuttaa Bootstrapin css-tiedosto less-tiedostoksi, jotta tarvittavat tyylimäärittelyt syöttökentille olisi saatu tuotua myös tyylioppaaseen. Tällöin Bootstrapin typografian määrittelyt olisivat kuitenkin sekoittaneet edelleen esimerkiksi uusia h1-otsikoita ja tämän takia päädyttiin tuomaan vain tarvittavat tiedostot määrittelyineen. Lisäksi on turhaa tuoda kaikkia Bootsrtapin tyylimäärittelyitä sovelluksen koodiin, sillä se kasvattaa sovelluksen tyylitiedostojen kokoa. Jatkossa

mahdollisesti tarvittavien Bootsrtapin tyylimärittelyiden käyttäminen onnistuu helposti tuomalla Bootstrapista haluttu less-tiedosto käyttöön.



Kuva 33. Rekisteröinti

Kuvassa 33 on nähtävillä miltä lomakkeen syöttökentät näyttivät ennen ja miltä ne nykyään näyttävät. Muutosta varten piti sovelluksen tyylitiedostosta style.less hakea esimerkiksi kuvassa näkyvien span- ja i-elementtien tyylimäärittelyt ja viedä ne uuteen forms.less-tiedostoon, josta niitä käyttää sekä sovellus että tyyliopas. Syöttökentät oli jo ennestään rakennettu pitkälti Bootstrapin tyylimäärittelyiden varaan, joita siihen saatiin esimerkiksi input-prepend-luokalta, joten nämä luokat pidettiin myös uusissa syöttökentissä. Syöttökenttää varten luotiin formInput niminen luokka, jonka avulla pystyttiin antamaan halutut tyylimääritteet syöttökentälle halutun ulkoasun mahdollistamiseksi. Uusia syöttökenttiä tehtäessä päätettiin myös samalla siirtyä käyttämään uusia ikoneita samasta Streamline-ikonifonttitiedostosta, josta käytetään sovelluksen muitakin ikoneita. Streamline-ikonifonttitiedostosta etsittiin vanhoja ikoneita vastaavat kuvat, kuten käyttäjähahmo, lukko sekä kirjekuori. Nämä nimettiin samalla periaatteella, mikä ikonien nimeämisessä on muutenkin käytössä sovelluksessa.

```
// Form input
//
// Basic inputfield to use in Ideapp. Replace icon depending where to use inputfield.
//
// Markup:
// <div class="control-group">
//      <div class="formInput input-prepend">
//          <span class="add-on"><i class="ideapp-envelope"></i></span>
//          <input type="email" id="username" value="" placeholder="Password" />
//      </div>
// </div>
//
// Styleguide 4.2
.formInput{
    >.add-on{
        padding: 12px;
        background-color: @white;
        .border-radius(5px,0px,0px,5px);
        line-height: 25px;
        clear: both;
            >i{
                font-size: 1.6em;
            }
        }>input {
            width: 160px;
            padding: 12px;
            .border-radius(5px,5px,0px,0px);
        }
}
```

Kuva 34. Syöttökentän määritteet

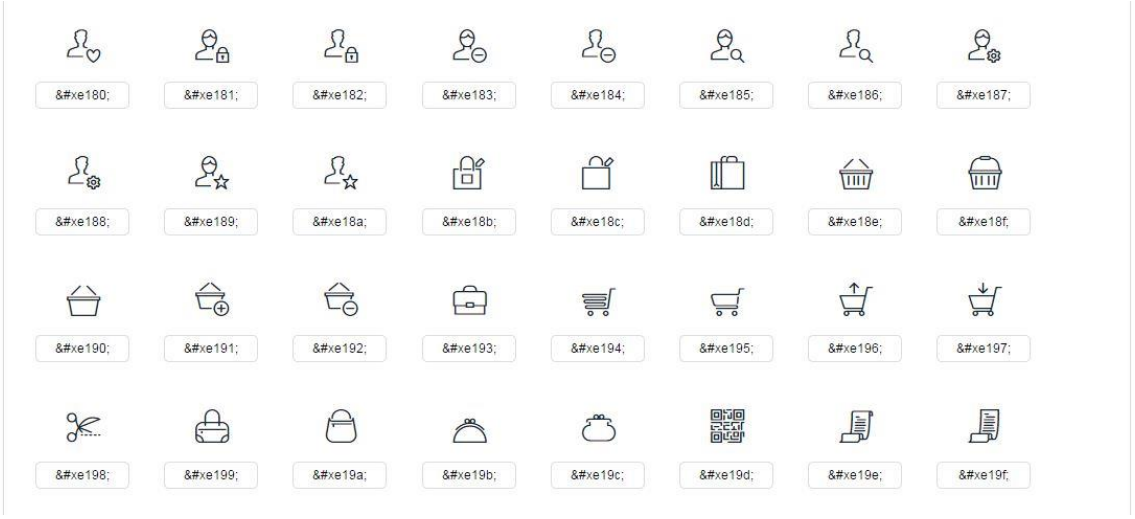
Kuvassa 34 on esitetty uudessa forms.less-tiedostossa oleva formInput -syöttökentän tyylimäärittelyt sekä nähtävillä on myös kuinka merkintä tapahtuu tyyliopasta varten. Vanhan syöttökentän span-elementillä olleet tyylimäärittelyt annettiin add-on-luokalle, jotta ne olisivat vahvemmat määrittelyt kuin span-elementille annetut ja tulisivat käyttöön. Jos määrittelyt olisi pidetty span-elementillä, olisivat Bootstrapin add-on-luokalle annetut tyylimääritteet yliajaneet esimerkiksi paddin-arvon eikä syöttökentistä olisi saatu saavutettu halutunlaista lopputulosta.

3.9 Ikonit

Ideapp:issa on käytössä Streamline-ikonifonttitiedosto, joka on kattava kokonaisuus vektorigrafiikalla toteutettuja ikoneita. Osa Streamline-ikonifonttipaketeista on ilmaisia ja osa maksullisia (Streamline 2015). Tästä tiedostosta sovellukseen oli aikaisemmin otettu käyttöön muutama ikoni kuten käyttäjän profiili-sivulle vievän painikkeen käyttäjähahmo sekä sivupalkin ”My rating”-tekstin vieressä oleva käyttäjähahmo tähdellä. Osa kuvakkeista tuli toisesta ikonitiedostosta, jonka yrityksen työntekijä oli itse tehnyt. Esimerkiksi kirjautumisen yhteydessä ollut lukko tuli tästä tiedostosta. Jotta toteutusta saatiin yhtenäistettyä, päätettiin vaihtaa kaikki ikonit tulemaan samasta ikonifonttitiedostosta. Täksi yhtenäiseksi

tiedostoksi valittiin Streamline-ikonifonttitiedosto, koska se on kattava kokonaisuus ikoneita ja mahdollisesti jatkon kannalta sieltä löytyy monia tarvittavia ikoneita. Ikonifontissa on myös se hyvä puoli, että koska ikonit ovat niin sanotusti fonttina, niitä pystyy helposti käsittelemään ja esimerkiksi muuttaa niiden kokoa. Ikonit on lisäksi toteutettu vektorigrafiikalla, joten ne säilyttävät terävyytensä skaalatessa niitä suuremmiksi verrattuna esimerkiksi normaaleihin png-kuviin.

Kuten muitakin tyylioppaaseen tulevia komponentteja varten, luotiin ikoneillekin oma icons.less -tiedosto, johon kerättiin ja lisättiin Ideapp:in ikonit. Koska ikoneita käytetään Streamline-ikonifonttitiedostosta, ikonit tuodaan tiedostoon niille annettujen arvojen mukaan.



Kuva 35. Kuvankaappaus Streamline-ikonifonttitiedostosta

Kuva 35 on kuvankaappaus Streamline-ikonifonttitiedostosta ja siitä nähdään, että esimerkiksi käyttäjähahmo-ikonin määritteenä on . Jotta ikoni saadaan sisällytettyä sovellukseen ja tyylioppaaseen, se tulee tuoda icons.less-tiedoston avulla käyttöön. Tämä onnistuu kuvan 36 osoittamalla tavalla.

```
.ideapp-hand-coin:before {
  content: "\e366";
}

.ideapp-treasure-chest-1:before {
  content: "\e4bb";
}

.ideapp-treasure-chest-2:before {
  content: "\e4bc";
}

.ideapp-user-star-2:before {
  content: "\e18a";
}
```

Kuva 36. Kuvankaappaus icons.less -tiedostosta

Kuvasta 36 voidaan nähdä esimerkkinä ikoneissa käytetty nimeämistapa, joka noudattaa samaa tyyliä, joka Ideapp:issa oli jo aikaisemmin käytössä ikoneilla. Ikonia varten icons.less -tiedostoon luodaan luokka, jonka nimeksi annetaan ”.ideapp-ikonin-nimi” ja lisämääreeksi before, joka sijoittaa ikonin elementin eteen. Tämä toimintatapa oli jo aikaisemmin käytössä sovelluksessa, joten samaa tapaa jatkettiin myös tyyliopasta tehdessä. Ikonin luokalle annetaan sisällöksi (content) Streamline-ikonifonttitiedostosta ikonia vastaavan määritteen neljä viimeistä arvoa. Icons-tiedosto ei sisällä ikoneille minkäänlaisia tyylimääritteitä vaan ikonit saavat tyylimääritteensä kuten kokonsa Ideapp:in style.less-tyylitiedostosta niistä kohdin, joissa ikoneita käytetään. Näin ollen ikonien less-tyylitiedosto eroaa hieman muista esimerkiksi painikkeiden less-tyylitiedostosta.

3.10 Otsake

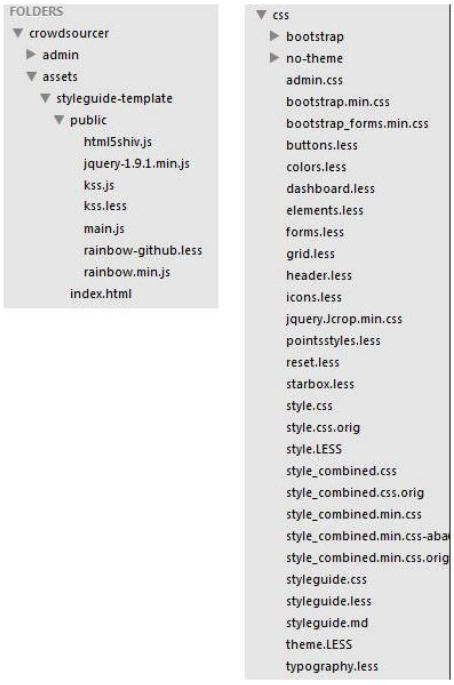
Otsake päätettiin irrottaa Ideapp:in style.less -tiedostosta ja sisällyttää se tyylioppaaseen yhdeksi elementiksi, jotta sen mahdollisia muutoksia olisi helpompi toteuttaa. Otsakkeen toteutus noudatti jo tutuksi tullutta kaavaa elementtien toteutuksesta ja tyylioppaaseen lisäämisestä. Ensimmäiseksi luotiin otsakkeelle oma less-tyylitiedosto, joka tuodaan käyttöön molempien sekä tyylioppaan että sovelluksen tyylitiedostoihin, kuten aikaisemminkin on toimittu. Otsake oli suurin rakenne, joka tässä vaiheessa sisällytettiin tyylioppaaseen. Sen toteuttaminen ei ollut sen vaikeampaa kuin muidenkaan tyylioppaaseen toteutettujen komponenttien toteuttaminen. Erona oli lähinnä se, että otsake rakentuu useasta komponentista ja näin ollen niiden kaikkien tulee ”sopia yhteen”. Otsakkeen siirtäminen omaan tyylitiedostoon oli samanlainen operaatio kuin muidenkin komponenttien kanssa, mutta otsakkeen kanssa sai olla tarkkana siitä, mitkä kaikki tyylimääritteet kuuluvat otsakkee-

seen. Otsakkeeseen liittyviä tyylimääritteitä on useampia kuin muissa aikaisemmin toteutetuissa elementeissä. Selaimen ”Inspect element”- työkalulla täytyi katsoa ja ymmärtää tyylimääritteiden periytyminen, jotta otsakkeesta sai oikeanlaisen rakenteen. Esimerkiksi line-heigth -arvo annettiin otsakkeelle samalla kun se siirrettiin omaan tyylitiedostoon, jotta otsake saa oikean arvon myös tyylioppaassa. Aikaisemmin otsake peri line-heigth -arvon body-elementiltä.

4 Lopputulos

Lopputuloksena opinnäytetyöstä syntyi Richenille tyyliopas heidän Ideapp-sovellukseensa. Koska tyyliopas on elävä dokumentti ja sen sisältö elää sovelluksen mukana, tyyliopas ei ”koskaan” ole valmis. Richenille tehty tyyliopas sisältää Ideapp:n tyylioppaan toteutushetkellä sisältämät komponentit, värit, ikonit sekä otsakkeen. Tyylioppaaseen ei kuitenkaan lähdetty toteuttamaan sovelluksen lista-elementtejä, sillä ne tulevat lähitulevaisuudessa muuttumaan kokonaan, joten niiden tekeminen tyylioppaaseen ja sitä kautta sovellukseen jätettiin Richenin työntekijöiden tehtäväksi.

Style-less -tiedosto pieneni noin 48 KB:hen ja koodirivejä jäi 2395 eli 275 riviä vähemmän kuin aikaisemmin. Sovelluksen tyylitiedosto pieneni siis noin 10%, mikä on hyvä muutos tyylitiedoston koossa ottaen huomioon, että siitä ei vielä poistettu aivan kaikkia sovelluksen elementtejä. Sovelluksen tyylitiedosto tulee vielä pienenemään jahka Richenin työntekijät suunnittelevat sovelluksen ulkoasua niiltä osin, joilta se tulee muuttumaan lähiaikoina. Esimerkiksi lista-elementtien poisto sovelluksen tyylitiedostosta tulee pienentämään tätä entisestään. Style.less -tiedostosta hävinneet koodirivit sijoitettiin uusiin tyylitiedostoihin. Uusia komponenttien tyylitiedostoja tuli yhteensä kuusi ja näiden lisäksi tyylioppaan oma tyylitiedosto. Tyylioppaalle on myös oma html-tiedosto sekä esimerkiksi styleguide.md -tiedosto, joka sisältää tyylioppaan yleiskatsauksessa olevan tekstin.



Kuva 37. Hakemistorakenne

Kuva 37 havainnollistaa Ideapp:n hakemistorakennetta tyylioppaan lisäämisen jälkeen projektiin. Hakemistorakenne toteutettiin Richenin työntekijöiden haluamalla tavalla ja uudet tiedostot sijoitettiin niihin hakemistoihin, jotka he kokivat olevan käytön kannalta parhaimmat hakemistot. Tyylioppaan tiedostot sijoitettiin assets-hakemistoon, josta löytyy muun muassa tyylioppaan index.html -tiedosto. Tyyliopasta varten tehdyt komponenttien omat less-tiedostot sijoitettiin css-hakemiston alle, jossa on myös muut sovelluksen tyylitiedostot sekä tyylioppaan oma tyylitiedosto.

Tyylioppaan ulkoasu pyrittiin pitämään mahdollisimman yksinkertaisena ja selkeänä. Ainoat muutokset alkuperäiseen KSS:n tyylioppaan pohjaan ovat lähinnä värimuutoksia taustavärissä ja fontin värissä. Lisäksi esimerkiksi ikoneille ja väreille tehtiin omat ”esittelylaatikonsa”, jotta ulkoasusta saisi näiden osalta mahdollisimman helpon ja ymmärrettävän. Näitä muutoksia lukuun ottamatta työssä käytettiin KSS:n normaalia tyyliopaspohjaa, koska se koettiin riittävän visuaaliseksi eikä nähty tarvetta sen ulkoasun suurempiin muutoksiin. Tyyliopas toimii useimmilla selaimilla kuten IE, Firefox ja Chrome. Tyylioppaan ulkoasua voi tarkastella liitteestä 4, joka sisältää kuvankaappauksina tyylioppaan.

Liitteenä oleva tyyliopas esittelee projektin lopputuloksena luodun tyylioppaan. Tyyliopas on jaettu kuuteen osioon, joista jokainen esittelee eri kategorian. Tyylioppaan alussa on lyhyt teksti tyylioppaasta ja sisällytetty linkki KSS:n dokumentaatiota varten KSS:n internetsivuille. Suurempia ohjeistuksia itse KSS:n dokumentaatioon ei koettu tarpeelliseksi tällä hetkellä, sillä yrityksen työntekijät ovat käyttäneet KSS:ää aikaisemmin ja dokumentaatio on siltä osin heille tuttu. Ensimmäinen varsinainen osio tyylioppaassa sisältää sovelluksen painikkeet. Liite numero viisi esittelee buttons.less -tiedostoa ja tämän rakennetta. Tästä osiosta muodostui laajin kokonaisuus tyylioppaassa. Liitteestä viisi voidaan nähdä kuinka osiossa on tehty painikkeille ”peruspainike”, jonka pohjalle muut painikkeet rakentuvat. Tämä peruspainike sisältää painikkeiden tyylimäärittelyistä muun muassa taustavärin ja vastaavat yleiset määrittelyt, jotka saadaan näin helposti jaettua kaikkien painikkeiden käyttöön. Tämän lisäksi painikkeet sisältävät yksilölliset määritteensä halutun ulkoasun saavuttamiseksi. Painikkeet toteutettiin sekä a- että button-elementeillä, jotta painiketta voidaan käyttää kaikissa tarvittavissa tapauksissa. Näin toimittiin esimerkiksi sen takia, että a-elementillä voidaan helposti toteuttaa linkki, jolloin saadaan halutun näköinen painike myös mahdolliseen linkkiin. Esimerkkien alla on a- ja button-elementtien toteuttamisen mahdollistava html-koodi, joista valitaan jompikumpi lisättäväksi html-pohjaan ja näin ollen tuomaan painikkeen käyttöön. Tämä painikkeiden toteutus sekä a- että button-elementillä niin, että ne näyttävät samoilta ei aina ollut helppoa, koska kyseisillä elementeillä on esimerkiksi valmiiksi tiettyjä valmiita tyylimääritteitä. Näistä eroavaisuuksista huo-

limatta pystyttiin tekemään samanlainen painike molemmilla elementeillä, ja molemmat ”versiot” käyttävät samoja tyylimääritteitä.

Toinen osio tyylioppaassa sisältää typografian, jossa on määritelty muun muassa päätason otsikoiden tyylit sekä normaalin tekstin tyyli. Sovelluksessa korvattiin joitakin otsikoiksi tehtyjä tekstejä näillä uusilla päätason otsikoilla. Tällä tavoin saatiin esimerkiksi sovelluksen ulkoasua yhtenäistettyä. Myös tulevaisuuden mahdollisia muutoksia varten ylläpito on helppoa, sillä muutos yhteen tiedostoon yhteen kohtaan, vaikuttaa useampaan kohtaan itse sovelluksessa. Kolmas osio tyylioppaassa sisältää sovelluksen värit ja esittelee ne. Liite numero kuusi esittelee colors.less-tiedostoa ja kuinka se on rakennettu. Tämän osion tyylitiedosto eroaa ikonien tyylitiedoston lisäksi muista osioista siinä, että nämä eivät sisällä varsinaisia tyylimääritteitä. Värien ja ikoneiden tyylitiedostojen avulla niiden ”komponentit” tuodaan sovellukseen käyttöön. Kuten liitteestä kuusi voidaan nähdä, osion tyylitiedosto sisältää värien hexadesimaaliarvot, jotka linkitetään less-muuttujien avulla käyttöön muualle sovellukseen. Esimerkiksi painikkeissa värit käyttävät näitä muuttujia ja painikkeen värin arvoksi voidaan näin ollen antaa @color-red. Tämä muutos helpottaa uusien värien lisäämistä ja vanhojen värien hallittavuutta ja muokattavuutta sovelluksessa. Esimerkiksi jos punaisen sävyä halutaan tulevaisuudessa muuttaa astetta tummenpaan, voidaan colors.less-tiedostoon tehdä muutos yhdellä hexadesimaaliarvon muutoksella ja muutos vaikuttaa tämän jälkeen heti kaikkiin elementteihin, joilla on @color-red niiden värimäärittelynä. Jatkossa siis voidaan tyylioppaasta valita kehittäessä haluttu väri ja käyttää sen less-muuttujaa tuomaan uuteen komponenttiin haluttu taustaväri. Värien esittelemistä varten tyylioppaaseen luotiin neliö, jonka taustaväriksi annetaan väri ja neliön pohjalla on värin less-muuttuja ilmaisemassa sitä määrittelyä, jolla kyseisen värin saa tuotua elementille. Värit jaettiin myös tyylioppaassa värien mukaan kuten punainen, ruskea ja harmaa. Nämä ”pääluokat” sisältävät kyseisten värien eri sävyt, jotka väreistä on käytössä.

Neljäs osio tyylioppaassa sisältää lomake-elementtejä. Tässä osiossa on tehty sovelluksessa käytetty syöttökenttä sekä esimerkiksi sisäänkirjautumisessa ja rekisteröinnissä käytetty syöttökenttä-elementti ikonilla. Myös sisäänkirjautumisessa käytetty jonkin virheellisen tiedon ilmaiseva lisäluokka, joka muodostaa syöttökentän ympärille punaisen reunan, on lisätty tähän osioon. Viides osio tyylioppaassa sisältää sovelluksen ikonit. Ikonien tyylitiedostoa on nähtävillä liitteessä numero seitsemän. Kuten värien kanssa, eroaa tämä osio hieman muista, siinä että tämäkään osio ei sisällä ikoneille tyylimäärittelyitä vaan ikonit pelkästään tuodaan käyttöön tämän avulla. Tyylimääritteet ikoneille annetaan paikkakohtaisesti sovelluksen tyylitiedostossa. Liitteestä seitsemän voidaan huomata ikonien käyttöönotto ja muun muassa se kuinka KSS:n syntaksista on jätetty pois ”Mar-

kup”, jolloin rivien 3-22 html-koodit eivät listaudu tyylioppaaseen näkyville. Html-koodien sisältämä tieto näytetään kuitenkin tyylioppaassa. Näin saadaan jätettyä tyylioppaasta tarpeetonta html-koodia pois, joka ei ole oleellista ikonien käyttämisen kannalta tyylioppaassa. Kuten väreille, tehtiin myös ikoneille tyylioppaaseen niiden esittelemistä varten oma esittelylaatikkonsa, joka sisältää ikonin ja tämän määrittelyn, jolla ikonin saa käyttöön. Viimeinen osio sisältää sovelluksen otsakkeen, jolla luotiin tyylioppaaseen oma osio.

Liite kahdeksan esittelee tyylioppaan tyylitiedoston. Tyylitiedoston alusta voidaan nähdä kuinka esimerkiksi muut less-tiedostot tuodaan käyttöön (import) tyylioppaaseen. Samalla tavalla ne tuodaan käyttöön myös sovellukseen tämän tyylitiedostossa. Liiteestä voidaan myös nähdä, että siinä on annettu KSS:n syntakseiden avulla tyylioppaan osioiden otsikot sekä näiden lyhyet alkutekstit. Tyylitiedosto sisältää myös ikoneita ja värejä varten tehtyjen yksinkertaisten ”esittelylaatikoiden” tyylimääritteet. Lisäksi siinä on muutettu tyylioppaan otsikon väriä halutuksi. Muut tyylimääritteet, joiden pohjalta KSS luo tyylioppaan ovat kss.less -tiedostossa, joka löytyy projektissa assets-hakemiston alta.

Uusia yleispäteviä komponentteja kokeiltiin sovelluksessa sekä otettiin siinä myös käyttöön sovelluksessa. Varsinkin painikkeiden osalta sekä väreiltään ja otsikoiltaan sovellus koki suurimman muutoksen kohti yhtenäisempää ulkoasua. Osa väreistä esimerkiksi oli vanhoja eikä niitä ollut käytössä enää missään, joten ne poistettiin koodista kokonaan. Osa väreistä korvattiin toisella, jotta ulkoasusta saataisiin yhtenäisempi. Esimerkiksi harvoin käytettyjä harmaan sävyjä korvattiin toisella harmaan sävyllä, jota käytettiin sovelluksessa jo ennestään useammassa kohdassa.

Richenin työntekijä Marcus Tallberg kommentoi lopputulosta seuraavasti:

> *”Tyylikirjaston (Styleguide) luominen olemassa olevaan palveluun on haastava tehtävä. Useimmiten suurimmat hyödyt toteutuksen osalta saadaan ottamalla tyylikirjasto käyttöön heti projektin ensimetreillä.*
> *Julius Juntusen luoma tyylikirjasto Ideapp-verkkopalveluun helpottaa tulevaisuudessa uusien toiminnallisuuksien luontia. Tyylikirjaston avulla uudet toiminnallisuudet saa toimivasti pirstoutettua pieniksi fragmenteiksi. Luotu tyylikirjasto otetaan osaksi kehitysprosessia ja otetaan sen avulla askelia kohti tyylikirjasto-vetoista kehittämistä.” (Tallberg 13.1.2015)*

Kuten Tallbergin kommentista voidaan todeta, tullaan tyyliopas ottamaan käyttöön Richenillä. Tallbergin mukaan tyyliopas tulee myös helpottamaan tulevaisuudessa uusien toiminnallisuuksien luontia. Kuten Tallberg toteaa, ollaan Richenillä siirtymässä tyyliop-

paan mukana kohti tyyliopas-vetoista kehittämisprosessia. Tämä toimintatapa vaatii yrityksen työntekijöiltä hieman totuttelua, jotta he saavat rutiinin tyylioppaan käyttöön kehityksessä. Nyt luotu tyyliopas on hyvä alku Richenin tyyli opas-vetoiseen kehitykseen ja se tarjoaa pohjan jatkaa kehitystä ja komponenttien lisäämistä tyylioppaaseen.

5 Pohdinta

Yhteenvetona tyylioppaista voitaisiin todeta, että hyvä tyyliopas auttaa työntekijöitä yhtenäisemmän ulkoasun luomisessa. Tyylioppaan tulisi myös nopeuttaa kehitystyötä, koska kehityksen komponentit on valmiiksi luotu ja tyyliopas tarjoaa komponentit helposti käyttöönotettaviksi. Näin ollen lopputuloksena tulisi olla paremmin toimiva sivu tai sovellus. Jotta tyylioppaasta saataisiin mahdollinen hyöty, sen tulisi olla helppolukuinen ja helposti ymmärrettävä. Tyylioppaan tulisi mahdollistaa yksinkertainen komponenttien ja rakenteiden uudelleenkäytettävyys yksinkertaisesti kehittäjän hakemalla tyylioppaasta haluttu komponentti ja kopioimalla se haluttuun kohtaan kehitteillä olevaan rakenteeseen. Parhaimmillaan tyylioppaan tulisi tarjota kehittäjälle tunne kuin hän rakentaisi rakennuspalikoista haluttua kokonaisuutta yksinkertaisesti valitsemalla halutut osat ja liittämällä niitä toisiinsa.

Opinnäytetyö ei aivan pysynyt alkuperäisessä aikataulussa vaan venyi lopulta noin kolmella viikolla alkuperäisestä aikataulusta, koska opinnäytetyön puolenvälin tienoilla saatu työ hankaloitti opinnäytetyön tekemistä vähentämällä huomattavasti muun muassa mahdollisuutta mennä Richenin tiloihin tekemään opinnäytetyötä ja saamaan vastauksia kysymyksiin. Opinnäytetyötä hankaloitti se, että tyylioppaan toteuttanut opiskelija ei ole ollut sovelluksen kehittämisessä mukana aikaisemmin. Tämä hankaloitti aluksi jonkin verran komponenttien etsimistä sovelluksesta sekä sovelluksen toiminnan ymmärtämistä. Esimerkiksi jotkin komponentit käyttävät javascriptiä tietyn toiminnan toteuttamiseen ja nämä on linkitetty yksilöllisen tunnisteen (id) avulla. Näissä tapauksissa esimerkiksi oli hankaluuksia välillä ennen kuin ymmärsi mitkä ”osat” aikaisemmasta komponentista tuli säästää, jotta se toimisi kuten ennen. Lähinnä ongelmana oli tarvittavien asioiden löytäminen eri tiedostoista. Esimerkiksi jotkin komponentit oli sijoitettu hogan-tiedostoihin, jotka renderöidään tarvittaessa sovellukseen, mutta jotkin komponentit, kuten otsake, oli toteutettu suoraan index.html -tiedostoon. Yrityksen työntekijät auttoivat onneksi mahdollisuuksien mukaan hahmottamaan sovelluksen rakennetta ja komponenttien löytämistä. Varsinkin siinä vaiheessa kun elementin html-koodi vaati tyylimääritteiden lisäksi muutoksia, tämä hidaste komponenttien löytämiseen sovelluksen tiedostoista tuli esille. Kaikki tarvittavat muutokset saatiin kuitenkin lopulta tehtyä. Työtä hankaloitti myös jonkin verran se, että sovelluksessa käytettävät less ja hogan sekä grunt eivät olleet tuttuja ennestään. Myös itse tyyliopas oli tuntematon asia ennen opinnäytetyön tekemistä. Näistä hidasteista huolimatta opinnäytetyö saatiin vietyä päätökseen ja Richen sai työstä kaipaamansa tyylioppaan Ideapp-sovellukseensa.

Opinnäytetyöstä toteutunut tyyliopas sisältää Ideapp:issa tällä hetkellä olevia komponentteja ja koska tyyliopas ”elää” koodin mukana se ei varsinaisesti koskaan ole ”valmis”. Sitä mukaa kun lisää komponentteja tehdään tai vanhoja päivitetään, tyyliopas elää muutoksen mukana. Richenin työntekijöille jää tyylioppaan ylläpitäminen tulevaisuudessa. Myös tällä hetkellä tyylioppaasta poisjätetyt elementit kuten lista-elementit, jotka tulevat päivittymään lähitulevaisuudessa, jäävät Richenin työntekijöiden toteutettaviksi. Tyylioppaan lisäksi suurin osa projektin työstä oli selkeyttää Ideapp-sovelluksen koodia ja sen rakennetta ja hakemistorakennetta. Tyylioppaan lisäksi siis Richen hyötyy opinnäytetyöprojektista siten, että sovelluksen rakenne ja tiedostot mahdollistavat tyyliopas-vetoisen sovelluskehityksen. Tämä tarkoittaa sitä, että suurin osa työstä kohti tyyliopas-vetoista sovelluskehitystä on jo tehty ja Richenin työntekijöiden työksi jää lähinnä uusien tarvittavien less-tiedostojen luominen uusille elementeille tai uusien komponenttien luominen jo olemassa oleviin less-tiedostoihin. Kokonaisuutena opinnäytetyöprojekti teki siis paljon muutakin kuin loi pelkän tyylioppaan Ideapp:in opinnäytetyönteon tekohetkellä olevista komponenteista.

Itse projektista ja tyylioppaan tekemisestä sekä KSS:n käytöstä opin kuitenkin paljon ja niistä on toivottavasti tulevaisuudessa hyötyä muissakin töissä. Lisäksi opinnäytetyötä tehdessä karttui tietoa ja kokemusta webkehittämisestä ja tyylitiedostojen ja html-tiedostojen toteutuksesta. Projektin pohjalta sain myös lisää itsevarmuutta alalta ja arvokasta kokemusta projektityöskentelystä. Lisäksi itseoppiminen korostui projektin aikana varsinkin esimerkiksi KSS:n kanssa ja tämä kasvatti luottamusta omaa osaamista ja tiedonhakua kohtaan. Lopputulokseen olen tyytyväinen ja toivon, että pääsen tulevaisuudessakin toteuttamaan vastaavanlaisia projekteja joko yksin tai ryhmässä.

Lähteet

Braun, B. 18.11.2013. Front-end Style Guides. Luettavissa: http://www.bryanbraun.com/2013/11/18/front-end-style-guides. Luettu:14.10.2014.

Feather, I. 18.5.2014. A Maintainable Style Guide. Luettavissa: http://engineering.lonelyplanet.com/2014/05/18/a-maintainable-styleguide.html. Luettu: 20.10.2014.

Gale, S. 21.12.1995. A Collaborative Approach to Developing Style Guides. Luettavissa: http://www.sigchi.org/chi96/proceedings/papers/Gale/srg_txt.htm.
Luettu 20.10.2014.

GitHub. 2014. Luettavissa: https://github.com/. Luettu 7.12.2014.

GitHub Styleguide. 2014. Luettavissa: https://github.com/styleguide. Luettu 22.10.2014.

Grunt. 2014. Luettavissa: http://gruntjs.com/. Luettu 16.10.2014.

Grunt asennus. 2014. Luettavissa: http://gruntjs.com/getting-started. Luettu 12.11.2014.

KSS 2014. Luettavissa: http://warpspire.com/kss/. Luettu 7.12.2014.

KSS syntaksi. 2014. Luettavissa: http://warpspire.com/kss/syntax/. Luettu 7.12.2014.

LESS 2014. Luettavissa: http://lesscss.org/. Luettu 6.11.2014.

LESS muuttujat. 2014. Luettavissa: http://lesscss.org/features/. Luettu 6.11.2014.

Neville, K. smashingmagazine.com. 21.7.2010. Designing Style Guidelines For Brands And Websites. Luettavissa: http://www.smashingmagazine.com/2010/07/21/designing-style-guidelines-for-brands-and-websites/. Luettu:13.1.2015.

Pattern Lab demo 2015. Luettavissa: http://demo.patternlab.io/. Luettu: 17.1.2015.

Pitkänen, P. Digitoday. 18.11.2014. Nettimiljonääri Sami Inkinen: Nyt on mahtava aika perustaa yritys. Luettavissa: http://www.digitoday.fi/tyo-ja-ura/2014/11/18/nettimiljonaari-sami-inkinen-nyt-on-mahtava-aika-perustaa-yritys/201416039/66. Luettu 19.11.2014.

Sanasto. Komponentti, Rakenne, Mallipohja. Luettavissa: https://github.com/resource/fabricator/wiki. Luettu:14.10.2014.

Streamlineicons 2015. Luettavissa: http://streamlineicons.com/. Luettu: 4.1.2015.

Sullivan, N. 19.8.2013. [JSConfUS 2013] Nicole Sullivan: Creating Living Style Guides. Katsottavissa: http://www.youtube.com/watch?v=lsaG-EJcu88. Katsottu: 10.11.2014.

Sullivan, N. 5.6.2013. Creating Living Style Guides to Improve Performance. Luettavissa: http://www.stubbornella.org/content/2013/06/05/creating-living-style-guides-to-improve-performance/. Luettu:10.11.2014.

Tallberg, M. 13.1.2015. Developer. Richen Oy. Sähköposti.

The style guide guide. Luettavissa: http://vinspee.me/style-guide-guide/. Luettu 14.10.2014.

WampServer 2014. Luettavissa: http://www.wampserver.com/en/ Luettu 16.10.2014.

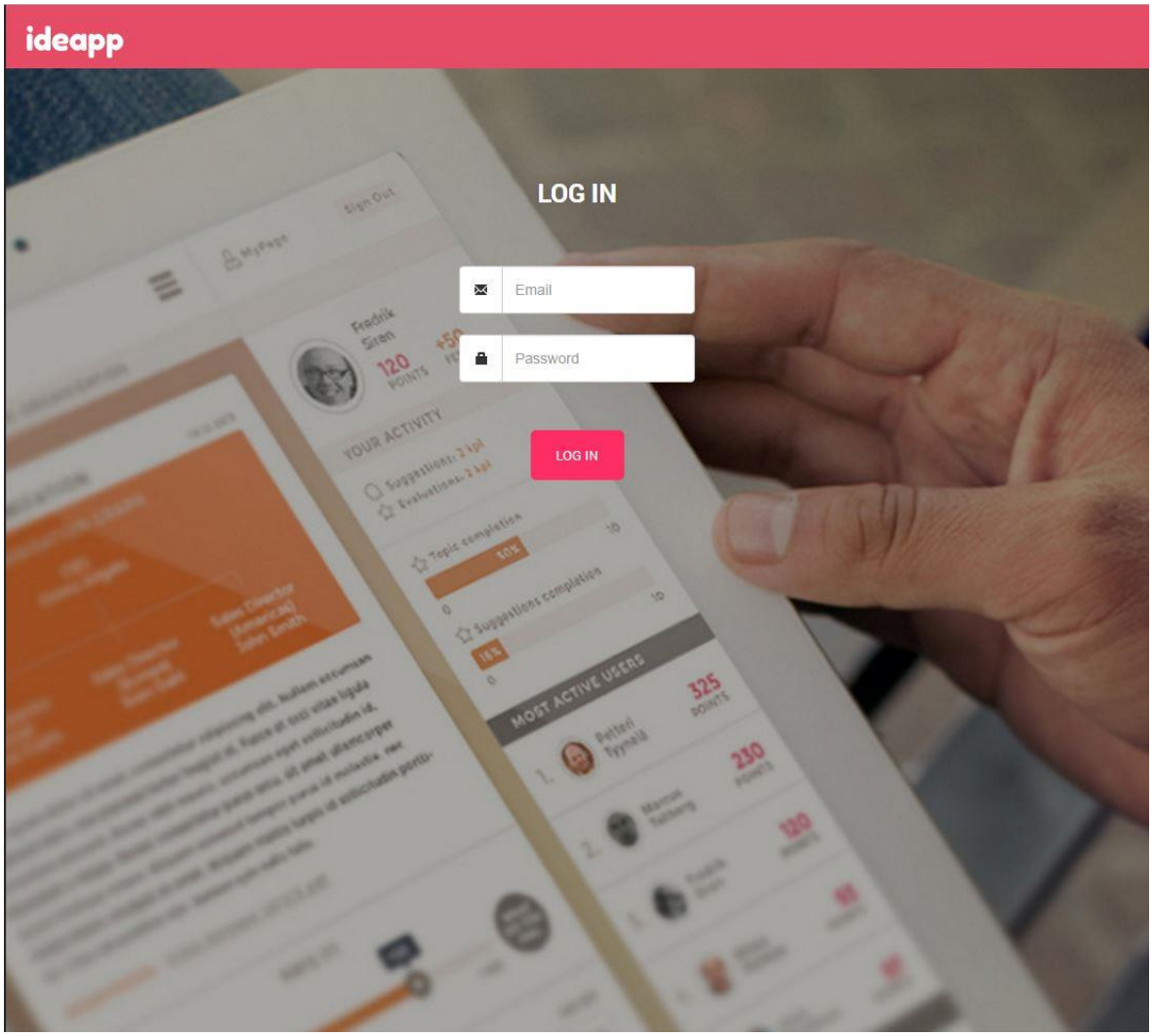
MailChimp tyyliopas 2014. Luettavissa: https://ux.mailchimp.com/patterns/. Luettu 17.10.2014.

Veikkaus tyyliopas 2014. Luettavissa: https://www.veikkaus.fi/web/web/release/sostyle/. Luettu: 22.12.2014.

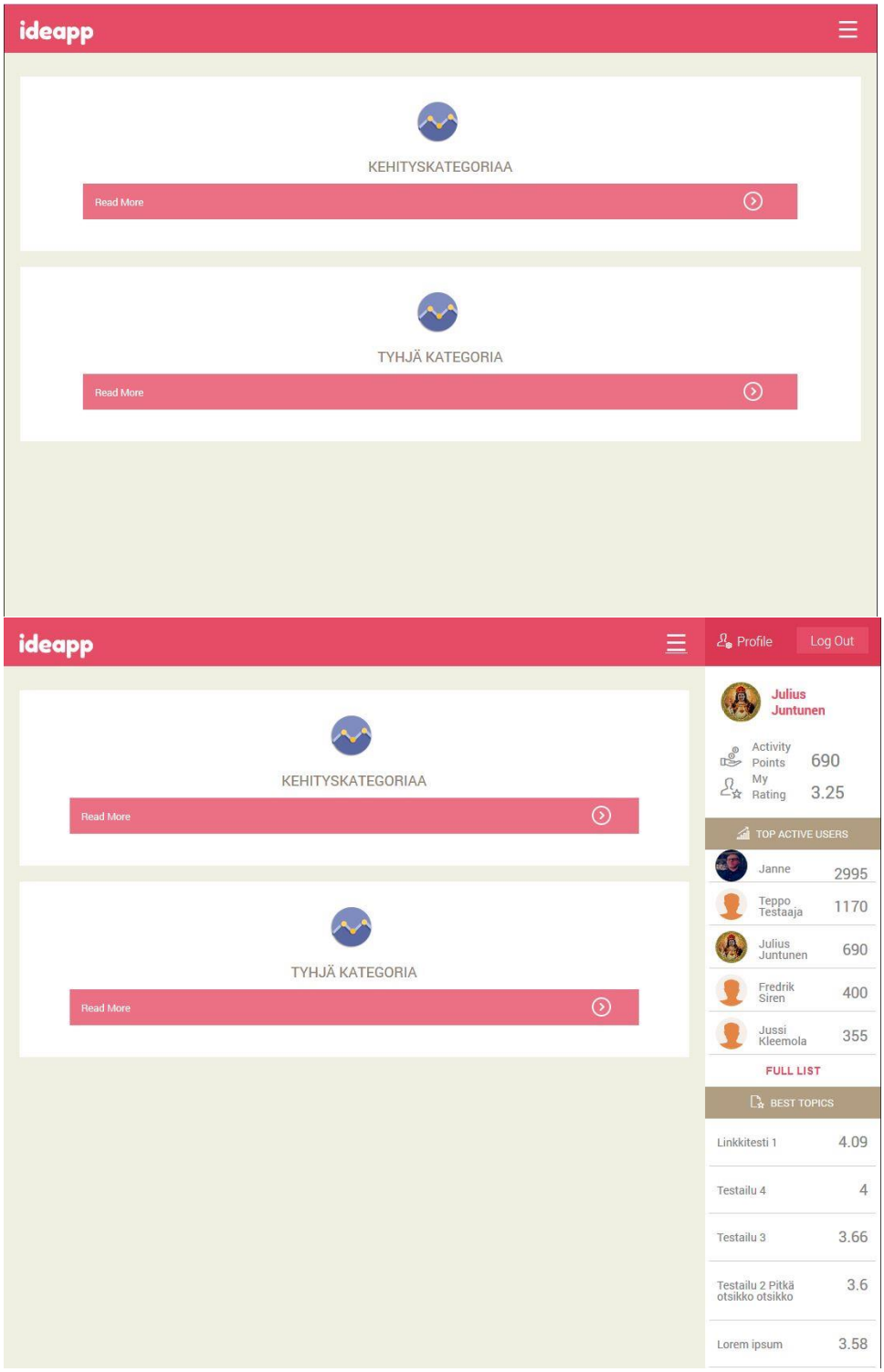
Wilson, C., E. Guidance on Style Guides: Lessons Learned. Uusintapainos Usability Interface, Vol 7, No. 4, April 2001. Luettavissa: http://www.stcsig.org/usability/newsletter/0104-style.html. Luettu 20.10.2014.

Liitteet

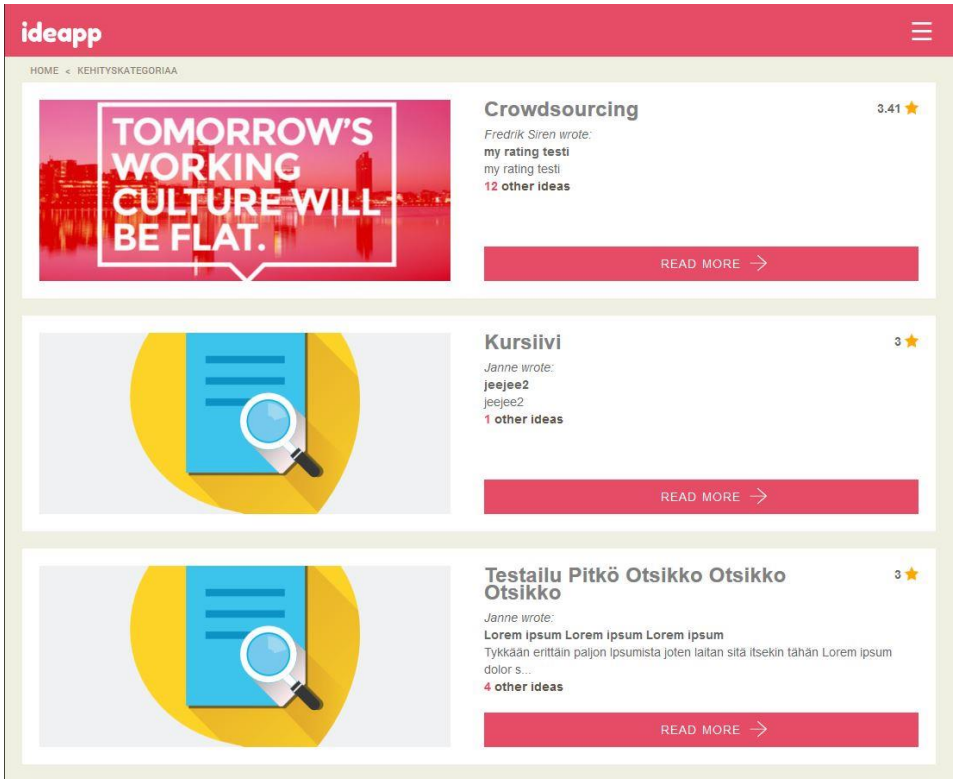
Liite 1. Ideapp sisäänkirjaus

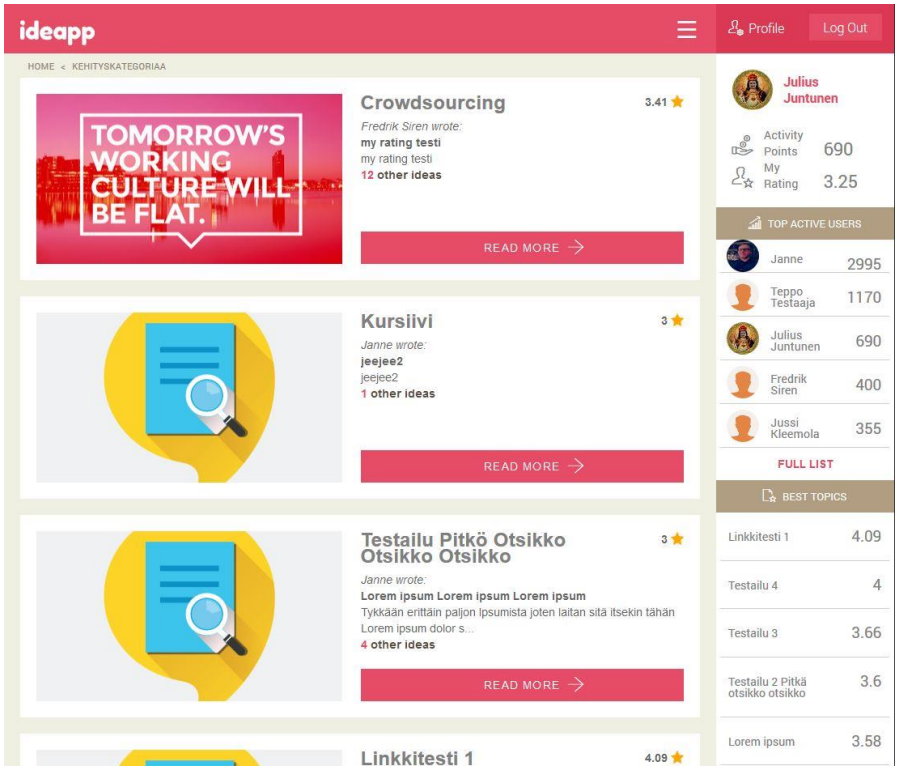


Liite 2. Ideapp kategoriat -näkymä sivupalkki kiinni ja auki



Liite 3. Ideapp aiheet -näkymä sivupalkki kiinni ja auki





Liite 4. Tyyliopas

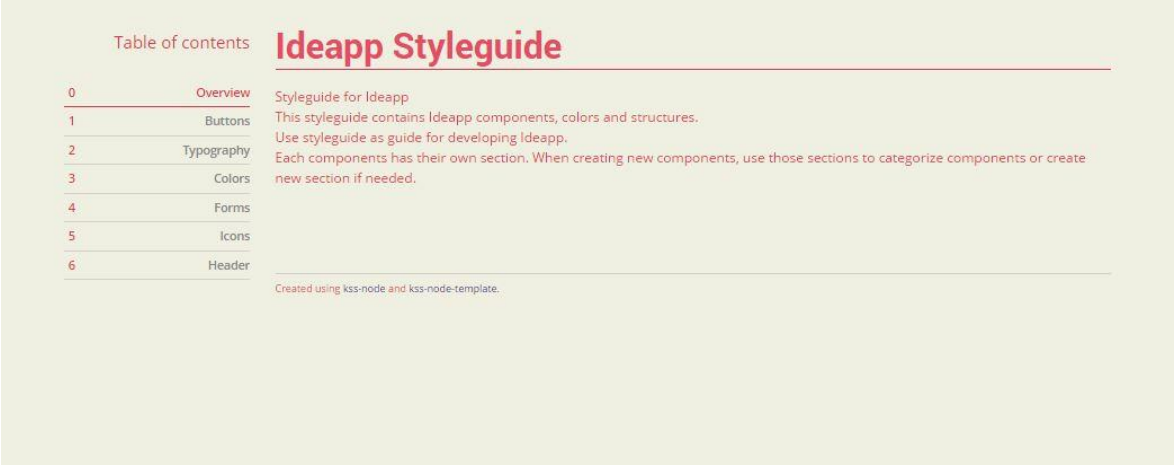


Table of contents

1 Buttons

Ideapp Button classes. Buttons can be done usign a or button element.

1.1 Basic Button

Basic action button used in ideapp.

Button Button

:hover Highlight the button when hovered.

Button Button

:active Active

Button Button

:visited Visited

Button Button

```
<a href="#" class="button {$modifiers}">Button</a>
<button class="button {$modifiers}">Button</button>
```

1.2 Open topic Button

Used in Ideapp Topic view

READ MORE →

READ MORE →

```
<a href="#" class="button openTopic {$modifiers}">Read more<span class="ideapp-arrow-32"></span></a>
<br>
<button class="button openTopic {$modifiers}">Read more<span class="ideapp-arrow-32"></button>
```

1.3 Form Button

Button used in Ideapp login and register.

LOG IN REGISTER

```
<a href="#" class="button formButton {$modifiers}">Log in</a>
<button class="button formButton {$modifiers}">Register</button>
```

1.4 Password change Button

Password change button used in Ideapp mypage password change.

CHANGE PASSWORD

CHANGE PASSWORD

```
<a href="#" class="button changepw {$modifiers}"><span>change password</span></a>
<br>
<button class="button changepw {$modifiers}"><span>change password</span></button>
```

Table of contents

CHANGE PASSWORD

CHANGE PASSWORD

```
<a href="#" class="button changepw {$modifiers}"><span>change password</span></a>
<br>
<button class="button changepw {$modifiers}"><span>change password</span></button>
```

1.5 Picture change Button

Picture change button used in Ideapp mypage.

```
<button class="button changepic ideapp-camera"></button>
```

1.6 Read more Button

Read more button used in Ideapp. Takes 85% of its parent element's width as its own width.

Read more

Read more

```
<a href="#" class="button readMore {$modifiers}"><span>Read more</span></a>
<br>
<button class="button readMore {$modifiers}"><span>Read more</span></button>
```

1.7 Sort Button

TOP TOP

```
<a href="#" class="button sort {$modifiers}">Top</a>
<button class="button sort {$modifiers}">Top</button>
```

1.8 Button with icon

Profile Profile

```
<a href="#" class="button iconButton {$modifiers}"><span class="ideapp-user-setting-2"></span><span>Profile<span></a>
<button class="button iconButton {$modifiers}"><span class="ideapp-user-setting-2"></span><span>Profile<span></button>
```

1.9 Button group

Use button group to get multiple buttons to same level.

Button Button Button

```
<div class="button-group">
        <a href="#" class="button">Button</a>
        <a href="#" class="button">Button</a>
        <a href="#" class="button">Button</a>
</div>
```

Created using kss-node and kss-node-template.

Table of contents

2 Typography

Ideapp typography.

2.1 H1

Size: 1.7em, Font-family: Roboto, sans-serif

This is heading 1

```
<h1>This is heading 1</h1>
```

2.2 H2

Size: 1.25em, Font-family: Roboto, sans-serif

THIS IS HEADING 2

```
<h2>This is heading 2</h2>
```

2.3 H3

Size: 1em

THIS IS HEADING 3

```
<h3>This is heading 3</h3>
```

2.4 Basic text

Basic text style for ideapp

Basic text Basic text Basic text Basic text Basic text Basic text Basic text

```
<div class="basicText">Basic text Basic text Basic text Basic text Basic text Basic text Basic text</div>
```

Created using kss-node and kss-node-template.

Table of contents

3 Colors

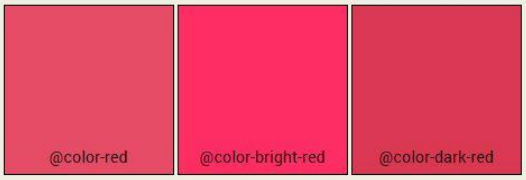
Colors used in Ideapp. To use color you want, use @color markdown on your css/less file.

3.1 Background colors

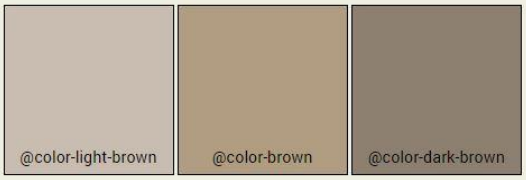
Basic background colors used in Ideapp
@color-white is used for sidebar
@color-light-grey1 is used for app background color
@color-light-black is used for body background color

3.2 Grey

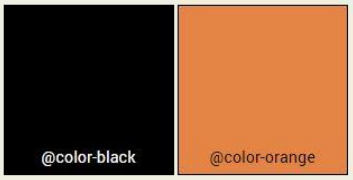
3.3 Red



3.3 Brown



3.4 Colors



Created using kss-node and kss-node-template.

Table of contents

4 Forms

Form elements used in Ideapp.

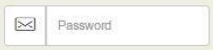
4.1 Input field

Basic input field to use in Ideapp.


```
<input class="inputField" type="text"  id="commentHeader_{{commentId}}" placeholder="Write your text here"/>
```

4.2 Form input

Basic inputfield to use in Ideapp. Replace icon depending where to use inputfield.

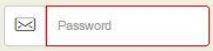



```
<div class="formInput input-prepend">
 <span class="add-on"><i class="ideapp-envelope"></i></span>
       <input type="email" id="username" value="" placeholder="Password" />
</div>
```

4.3 Inputfield with error

Error class given by javascript if inputfield doesn´t have right value.

.error




```
<div class="control-group">
              <div class="formInput input-prepend">
                     <span class="add-on"><i class="ideapp-envelope"></i></span>
                     <input class="error" type="email" id="username" value="" placeholder="Password" />
              </div>
</div>
```

Created using kss-node and kss-node-template.

Table of contents

5 Icons

Icons used in Ideapp. Use icon name as class

```
<div class="ideapp-user"></div>
```

5.1 Icons

 ideapp-treasure-chest-1	 ideapp-treasure-chest-2	 ideapp-user	 ideapp-chat-1
 ideapp-user-setting-2	 ideapp-user-star-2	 ideapp-bubble-star-1	 ideapp-file-favorite-2
 ideapp-signal-high	 ideapp-chart-up	 ideapp-arrow-32	 ideapp-arrow-33
 ideapp-arrow-34	 ideapp-arrow-68	 ideapp-hand-coin	 ideapp-piggy-bank
 ideapp-award-2	 ideapp-camera	 ideapp-envelope	 ideapp-lock

Created using kss-node and kss-node-template.

Table of contents

6 Header

Ideapp header

6.1 Header

Ideapp header


```
<div class="container-fluid" id="header">
   <div class="row-fluid custom-header-style">
       <div class="ideapp_content">
           <a id="logo" href="/"></a>
           <a href="#" id="collapse">&#9776;</a>
       </div>
       <div class="ideapp_sidebar collapsable authactionarea">
         <div id="controls">
           <span class="button-group">
             <a href="/dashboard/myprofile" id="mypage" class="button iconButton"><span class="ideapp-user-setting-2"></span><span data-i18
             <a href="#" class="button" id="logOut" data-i18n="index.logout">Log Out</a>
           </span>
         </div>
       </div>
   </div>
</div>
```

Created using kss-node and kss-node-template.

Liite 5. Buttons.less

buttons.less

```
// Basic Button
//
// Basic action button used in ideapp.
//
// Markup:
// <a href="#" class="button {$modifiers}">Button</a>
// <button class="button {$modifiers}">Button</button>
//
// :hover       - Highlight the button when hovered.
// :active      - Active
// :visited     - Visited
//
// Styleguide 1.1
.button{
    -webkit-appearance:none;
    -moz-appearance:none;
    appearance:none;
    background: @color-red;
    font-family: @basicFont;
    padding: 10px;
    border: 0px solid @color-red;
    text-align: center;
    color: @color-white;
    cursor: pointer;
    font-size: 1em;
    text-decoration: none;

    &:hover {
        .button-colorize(@color-dark-red);
        color: @color-white;
        text-decoration: none;
    }
    &:active{
        color: @color-white;
        text-decoration: none;
    }
    &:visited{
        color: @color-white;
        text-decoration: none;
    }
}

// Open topic Button
//
// Used in Ideapp Topic view
//
// Markup:
// <a href="#" class="button openTopic {$modifiers}">Read more<span class="ideapp-arrow-32"></sp
// <br>
// <button class="button openTopic {$modifiers}">Read more<span class="ideapp-arrow-32"></button
//
// Styleguide 1.2
.openTopic {
    display: block;
    letter-spacing: 1.2px;
    padding: 10px 0 10px 0;
    text-align: center;
    text-transform: uppercase;
    width: 100%;
        span{
            display: inline-block;
            font-size: 1.3em;
            margin-left: 10px;
            position: relative;
            top: 4px;
    }
}

// Form Button
//
// Button used in Ideapp login and register.
//
// Markup:
// <a href="#" class="button formButton {$modifiers}">Log in</a>
// <button class="button formButton {$modifiers}">Register</button>
//
// Styleguide 1.3
.formButton {
        background: @color-bright-red;
        .border-radius(5px,5px,5px,5px);
        border: @color-bright-red;
        padding:15px 25px 15px 25px;
        text-transform: uppercase;
    }
```

Liite 6. Colors.less

colors.less

```
// Background colors
//
// Basic background colors used in Ideapp <br>
// <span class="text-black">@color-white</span> is used for sidebar <br>
// <span class="text-black">@color-light-grey1</span> is used for app background color <br>
// <span class="text-black">@color-light-black</span> is used for body background color <br>
//
// <div class="color-box bg-white"><span>@color-white</span></div>
// <div class="color-box bg-light-grey1"><span>@color-light-grey1</span></div>
// <div class="color-box bg-light-black"><span class="dif">@color-light-black</span></div>
//
// Styleguide 3.1

@color-white : #fff;
@color-light-grey1 : #eff0e2;
@color-light-black:#282828;

.bg-white{background-color: @color-white;}
.bg-light-grey1{background-color: @color-light-grey1;}
.bg-light-black{background-color: @color-light-black;}

// Grey
//
// <div class="color-box bg-light-grey"><span>@color-light-grey</span></div>
// <div class="color-box bg-light-grey2"><span>@color-light-grey2</span></div>
// <div class="color-box bg-grey"><span>@color-grey</span></div>
// <div class="color-box bg-dark-grey"><span>@color-dark-grey</span></div>
//
// Styleguide 3.2

@color-light-grey : #f2f2f2;
@color-light-grey2 : #d4cfcb;
@color-grey : #808080;
@color-dark-grey: #666666;

.bg-light-grey{background-color: @color-light-grey;}
.bg-light-grey2{background-color: @color-light-grey2;}
.bg-grey{background-color: @color-grey;}
.bg-dark-grey{background-color: @color-dark-grey;}

// Red
//
// <div class="color-box bg-red"><span>@color-red</span></div>
// <div class="color-box bg-bright-red"><span>@color-bright-red</span></div>
// <div class="color-box bg-dark-red"><span>@color-dark-red</span></div>
//
// Styleguide 3.3

@color-bright-red : #FE2E64;
```

Liite 7. Icons.less

buttons.less

```
// Icons
//
// <div class="icon-box ideapp-treasure-chest-1"><span>ideapp-treasure-chest-1</span></div>
// <div class="icon-box ideapp-treasure-chest-2"><span>ideapp-treasure-chest-2</span></div>
// <div class="icon-box ideapp-user"><span>ideapp-user</span></div>
// <div class="icon-box ideapp-chat-1"><span>ideapp-chat-1</span></div>
// <div class="icon-box ideapp-user-setting-2"><span>ideapp-user-setting-2</span></div>
// <div class="icon-box ideapp-user-star-2"><span>ideapp-user-star-2</span></div>
// <div class="icon-box ideapp-bubble-star-1"><span>ideapp-bubble-star-1</span></div>
// <div class="icon-box ideapp-file-favorite-2"><span>ideapp-file-favorite-2</span></div>
// <div class="icon-box ideapp-signal-high"><span>ideapp-signal-high</span></div>
// <div class="icon-box ideapp-chart-up"><span>ideapp-chart-up</span></div>
// <div class="icon-box ideapp-arrow-32"><span>ideapp-arrow-32</span></div>
// <div class="icon-box ideapp-arrow-33"><span>ideapp-arrow-33</span></div>
// <div class="icon-box ideapp-arrow-34"><span>ideapp-arrow-34</span></div>
// <div class="icon-box ideapp-arrow-68"><span>ideapp-arrow-68</span></div>
// <div class="icon-box ideapp-hand-coin"><span>ideapp-hand-coin</span></div>
// <div class="icon-box ideapp-piggy-bank"><span>ideapp-piggy-bank</span></div>
// <div class="icon-box ideapp-award-2"><span>ideapp-award-2</span></div>
// <div class="icon-box ideapp-camera"><span>ideapp-camera</span></div>
// <div class="icon-box ideapp-envelope"><span>ideapp-envelope</span></div>
// <div class="icon-box ideapp-lock"><span>ideapp-lock</span></div>
//
// Styleguide 5.1

.ideapp-hand-coin:before {
  content: "\e366";
}

.ideapp-treasure-chest-1:before {
  content: "\e4bb";
}

.ideapp-treasure-chest-2:before {
  content: "\e4bc";
}

.ideapp-user-star-2:before {
  content: "\e18a";
}

.ideapp-bubble-star-1:before {
  content: "\e0d6";
}

.ideapp-signal-high:before {
  content: "\e101";
}

```

Liite 8. Styleguide.less

styleguide.less

```
@import "reset.less";
@import "elements.less";
@import "theme.less";
@import "bootstrap/variables.less";
@import "bootstrap/mixins.less";
@import "bootstrap/grid.less";
@import "bootstrap/forms.less";

// Buttons
//
// Ideapp Button classes. Buttons can be done usign a or button element.
//
// Styleguide 1
@import "buttons.less";
// Typography
//
// Ideapp typography.
//
// Styleguide 2
@import "typography.less";
// Colors
//
// Colors used in Ideapp.
// To use color you want, use <span class="text-black">@color</span> markdown on your css/less f
//
// Styleguide 3
@import "colors.less";
// Forms
//
// Form elements used in Ideapp.
//
// Styleguide 4
@import "forms.less";
// Icons
//
// Icons used in Ideapp. Use icon name as class
//
// Markup:
// <div class="ideapp-user"></div>
//
// Styleguide 5
@import "icons.less";
// Header
//
// Ideapp header
//
// Styleguide 6
@import "header.less";


.color-box{
    border: 1px solid @color-black;
    display: inline-block;
    height: 200px;
    margin-bottom:5px;
    width: 200px;
    position: relative;
    word-wrap: break-word;
        >span{
            bottom: 0;
            color: @color-black;
            font-family: @basicFont;
            text-align: center;
            font-size: 1.2em;
            height: 40px;
            left:0;
            line-height: 40px;
            position: absolute;
            width: 100%;
    }.dif{color:@color-white;}
}

```

```
.icon-box{
    color: @color-black;
    border: 1px solid @color-red;
    display: inline-block;
    font-size: 80px;
    margin-bottom: 5px;
    padding: 60px;
    position: relative;
    text-align: center;
        >span{
            border-top: 1px solid @color-red;
            bottom: 0;
            font-family: @basicFont;
            font-size: 0.2em;
            height: 40px;
            left:0;
            line-height: 40px;
            position: absolute;
            width: 100%;
    }
}

h1{line-height: normal;}

#ideapp-styleguide{
    color: @color-red;
}
.text-black{
    color: @color-black;
}
```